2013

Strategy for Health Monitoring and Fault Detection in Heavy-Duty Diesel Engines

Aniketjayant Vagha
Purdue University, aniketvagha@gmail.com

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Aniketjayant Vagha

Entitled
Strategy for Health Monitoring and Fault Detection in Heavy-Duty Diesel Engines

For the degree of Master of Science in Mechanical Engineering

Is approved by the final examining committee:

Peter H. Meckl
Chair

Galen B. King

Kristofer B. Jennings

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Peter H. Meckl

Galen B. King

Approved by: David C. Anderson 10/18/2013
Head of the Graduate Program Date

STRATEGY FOR HEALTH MONITORING AND FAULT DETECTION IN HEAVY-DUTY DIESEL ENGINES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Aniket Jayant Vagha

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Mechanical Engineering

December 2013

Purdue University

West Lafayette, Indiana

To my family

for

their unwavering support

ACKNOWLEDGEMENTS

A sense of accomplishment is always punctuated with an immense sense of gratitude towards the people who make the feat what it is! I am extremely fortunate to have worked on this research with such people and would like to extend my gratitude to them.

I would like to extend my sincere gratitude, and in the same breath, towards all of my research committee members – Peter Meckl, Galen King and Kristofer Jennings. They trusted me without any reservation as I took the plunge, and ensured that I felt secure and independent in thought and actions throughout the course of this work. The scientific brainstorming interjected with spontaneous humor made the weekly meetings great fun! Needless and clichéd though it may sound, I wouldn't be writing this document without their support.

I convey my earnest gratefulness to Cummins Inc. and its representatives for their patience and support throughout the project. Joseph Gahimer and Don Lakin helped us ease through the otherwise challenging initial stages of the work. Ryan Edgecomb and Tom Grana of HDE were an integral part of all the experimental and developmental work and I cannot thank them enough for sharing their expertise and vast experience in the field of engine performance and servicing.

It was an absolute privilege to have been a part of the Herrick Labs' family. The goodness and the congenial atmosphere at Herrick always ensured high spirits. As the laboratory braces itself to move to a plusher, swankier location, I wish them many more wonderful years ahead!

I would, specially mention Ben Warman, for guiding a naïve me through the first set of technical hurdles. Finally, my research buddies – Sai Shirsikar, Ashish Vora and Nishi Railkar - for making the last almost two years worthy to cherish for a long, long time!

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

LIST OF SYMBOLS

x	n-dimensional random vector
ϕ	Population under study
p	Probability distribution of a population
c( )	Cost of misclassification
γ	Prior probability
μ_c	Mean of class ‘c’
Σ_c	Covariance matrix for class ‘c’
N_c	Number of observations in class ‘c’
$\delta(x)$	Discriminant function
α	SLDA coefficients
$J(\alpha)$	Fisher’s criterion
S_b	Between-class variance
S_w	Within-class variance
λ_1	Penalty on L1-norm
λ_2	Penalty on L2-norm

LIST OF ABBREVIATIONS

CIT	Compressor Inlet Temperature
DOC	Diesel Oxidation Catalyst
DOT	Department of Transportation
DPF	Diesel Particulate Filter
ECM	Engine Control Module
EGR	Exhaust Gas Recirculation
EPA	Environmental Protection Agency
FDD	Fault Detection and Diagnosis
FDI	Fault Detection and Isolation
FMEA	Failure Mode Effect Analysis
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
MI	Mutual Information
MIL	Malfunction Indicator Lamp
MQT	Matching Quality Threshold
NaN	Not-a-Number
NN	Neural Network

OBD	On-Board Diagnostics
OLS	Ordinary Least Squares
ROC	Receiver Operating Characteristic
RR	Removing Repeats
RSS	Residual Sum of Squares
SCR	Selective Catalytic Reducer
SLDA	Sparse Linear Discriminant Analysis
SVM	Support Vector Machine
VGT	Variable Geometry Turbine

ABSTRACT

Vagha, Aniket Jayant. M.S.M.E., Purdue University, December 2013. Strategy for Health Monitoring and Fault Detection in Heavy-Duty Diesel Engines. Major Professors: Peter Meckl and Galen King, School of Mechanical Engineering.

Performance tuning, health monitoring, fault diagnosis etc. are important aspects of testing a pre-production engine out in the field. To fulfill these tasks, a robust data-based strategy for detecting anomalies in diesel engines is proposed in this work. Such a strategy separates the healthy data from outlying, anomalous trends of faulty data. The data classifier used here is based on fundamental principles of statistical learning and derived from linear discriminant analysis.

Further efforts to improve the classification performance led to the finding that steady state data makes for a more accurate classification of the working conditions of individual trucks. Hence an algorithm to extract steady data is suggested. After achieving nearly 100% accuracy for classifying data with one fault, this work is naturally extended to handle multiple trucks and the associated faults. Subsequently, a two-fault classifier using trucks belonging to distinct fleets and ratings, and a three-fault classifier using trucks belonging to the same family were devised. It was observed that data of trucks with similar power ratings and calibrations is more easily classified. The efforts towards clustering the healthy

data of three different trucks together and separating it from the clusters of faulty data were successful. This was achieved at an acceptable accuracy of greater than 90%.

The multi-fault, multi-class allocation scheme was validated using five completely new, healthy sets of data. These trucks contained engines with similar hardware and calibrations. It was observed that the data of all but one trucks were grouped in the healthy cluster. That one outlying truck, on further investigation, yielded unusually high values of crankcase pressure as compared to the other four. This established the validity of the technique in characterizing healthy data and identifying anomalous behavior of the on-field trucks.

CHAPTER 1. INTRODUCTION

1.1 Introduction

The transportation industry is not a mere means of movement but a system that drives the economy of a nation. An essential component of this huge infrastructure is a truck or a trucking company. Trucks, because of their ability to scale the length and breadth of the country, have become a preferred means of transport of goods and services. A desire to increase productivity and efficiency of heavy-duty trucks has led to many technological advancements. However, when it comes to the heavy-duty engines of these trucks, designs and innovations are inspired by stringent measures to control emissions and reduce fuel consumption.

In August 2011, President Barack Obama stood in support of a national policy to increase fuel efficiency and decrease greenhouse gas emissions from medium and heavy-duty trucks. These standards, developed by the Department of Transportation (DOT) and Environmental Protection Agency (EPA) in close coordination with the leading companies and manufacturers, mandate up to 20% reduction in fuel consumption and greenhouse gas emissions by 2018 [1]. This comes on top of the strict EPA regulation, pursuant to Clean Air Act Amendments of 1990, which requires the

manufacturers to install the newest HD-OBD (Heavy-duty On-board Diagnostics) systems on their heavy-duty engines [2].

The HD-OBD system monitors virtually every component that can affect the emission performance of the vehicle and assists repair technicians in diagnosing and fixing problems with the computerized engine controls. If a problem is detected, the HD-OBD system illuminates a warning lamp on the vehicle instrument panel to alert the driver. This warning lamp typically contains the phrase 'Check Engine' or 'Service Engine Soon' [3]. Most faults or anomalous behaviors, however, do not appear all of a sudden. They develop over a much longer span of time, building slowly within a system of the engine and gradually affecting the performance of connected subsystems. It is only after a preset threshold is crossed that the 'Check Engine' lamp or the Malfunction Indicator Lamp (MIL) lights up.

A classic example is building up of soot layers in the diesel particulate filter. It is only after the back pressure crosses the OBD mark and stays there or higher for a certain span of time that the MIL flashes. However, system analysis studies have shown that the back pressure does not jump over the threshold in a short time but gradually increases over time. Various system level plots of important relevant variables manifest this. Systems and performance engineers, therefore, are in constant need of efficient tools to aid their analysis-led design.

1.2 Motivation and Research Impact

Emissions and efficiency are the two most trending words in vehicle or parts manufacturing industries. The current strict benchmarks for exhaust emissions and fuel

consumption will be tightened in the next few years. Heavy-duty engine manufacturers must achieve this without compromising on performance, reliability and flexibility and ensure a long, fault-free life and lasting customer satisfaction. Hence, on-field validation of emission models and combustion recipes in pre-production engines is of utmost importance. Typically, heavy-duty engines are subjected to a few thousands to tens of thousands of hours of testing depending on the application. After-treatment validation, powertrain assessment, extreme weather effects and OBD checks are some of the important checkboxes to be ticked off during a field test. One such vital aspect of field testing is intensive data monitoring – keeping a watch over healthy behavior of trucks and identifying any anomalous, faulty trends.

The volume of data obtained from the field tests is huge. This data needs to be effectively segregated and represented in a form interpretable to the engineer. System engineers, based on their own experience and understanding, visually classify the good behavior from the bad behavior through various system-level plots. The manual analysis that follows takes up a significant commitment on the part of the engineering personnel, time and money. It was necessary to automate this analysis by reducing or completely eliminating human intervention without compromising the quality set by an engineer's ability to distinguish between healthy and faulty through prior experience and learning. This was the primary motivation of the work accounted in the following chapters.

The development of such a tool will immensely aid the engineers who spend hours analyzing the data. Such a tool will help reduce the costs incurred via servers running day

and night. Statistical data can thereby be utilized to develop a self-learning model and potentially aid the OBDs to perform more efficiently, reducing the instances of false positives.

1.3 Research Goals

The statement of the problem and its related background help set up our research goals. They are:

- To isolate the data representing anomalous behavior and characterize the data representing healthy behavior as distinct clusters.
- To use this knowledge to allocate the newly arriving test data into appropriate groups.

1.4 Previous Work

Our current goals, objectives and methodology have all evolved from almost a decade of related work in the group. Research on Fault Detection and Diagnostics (FDD) methods began as early as 2002 with model-independent clustering algorithms within an information theoretic framework. The work was directed towards identifying a lower dimensional input space from a wide array of parameters employing Mutual Information (MI) as the measure of correlation between the signals [4]. These signals would characterize the faulty conditions in the form of disjoint clusters – one cluster per fault. Within the premises of data-based fault detection, methods based on statistical modeling of associated symptoms were explored. A novel histogram-based method was formulated

to approximate non-parametric probability density functions [5]. One of the methods focused on healthy-faulty discrimination of engine states trained with data from a minimal number of signals. A second method aimed to statistically model the nominal behavior of the signal associated with a particular symptom [6]. Techniques such as non-linear time series analysis and local regression were used to monitor the health of charge air-cooler in diesel engines [7]. This initial work, which used both lab-based and field-tested data, laid the foundation strategies for physics-based and process-history based diagnostics.

Within the premises of process history-based parametric approaches, Sparse Linear Discriminant Analysis (SLDA) was identified as a potent tool to achieve input subset selection, dimension reduction and classification all in one step [8], [9]. Such a setup requires information-rich healthy and faulty data (single fault at a time) for training the classifier. Since these data sets could have been collected in totally different ambient conditions with trucks operating at different duty cycles, a potential selection bias would have been induced in the data. A novel, non-parametric data matching approach was devised to tackle this problem [10]. SLDA and data matching have directly motivated the work that is further documented in this thesis.

1.5 Organization of the Document

This thesis continues with Chapter 2, which presents an introductory account of the entire spectrum of fault detection and diagnosis methods. Some of the more popular techniques have been reviewed in more detail to provide the reader with better insight. Chapter 3 discusses discriminant analysis and lays the founding concepts and supporting mathematics

for the approach used in this work. Chapter 4 provides details of data setup in MATLAB, data preprocessing techniques like data matching, steady state analysis etc. This chapter also presents the results for an individual truck case wherein data of one fault in each truck was classified individually. Chapter 5 extends this solution to two and three trucks, respectively, put together in a binary (two-class) classifier. Chapter 6 deals with a multi-fault, multi-class classification where all faults are treated as different classes from one another. Chapter 7 furnishes the concluding remarks and scope for future work. This document also contains an appendix presenting a study of the operating regimes represented by the data and their influence on the performance of the model. A second appendix contains the relevant Matlab programs.

CHAPTER 2. REVIEW OF LITERATURE

Vast resources of literature are available to devise solution strategies for problems such as the one described in the previous chapter. Before delving into our methodology, it is imperative to shed some light on some of the commonly used methods for fault detection and diagnosis. Key aspects of these methods contribute to choosing a technique or a combination of techniques to present a feasible solution. This chapter focuses on some basic concepts and some popular procedures in fault detection and diagnostics.

2.1 Fault, Fault-Symptom Relation

At the outset, before perusing the extensive literature, it is useful to define a seemingly abstract idea of a fault. A fault is an abnormality in the routine behavior of the system. In the words of Rolf Isermann, "*it is an unpermitted deviation of at least one characteristic feature of the system from the acceptable, usual, standard condition"* [11]. Faults manifest themselves in the form of various effects on the system. These effects are called symptoms. Restating Isermann, "*the effect of a fault in a component can be observed through symptoms. Thus, faults cause symptoms"* [11]. A single fault can lead to many a symptom. A single symptom can be a result of multiple faults. Thus, there exists

a complex many-to-many correspondence between faults and symptoms [8]. Tools like Failure Mode Effect Analysis (FMEA) and fault – symptom tree help pinpoint a precise subset of fault symptom relations [3].

2.2 Fault Detection and Diagnosis (FDD)

Various strategies for fault detection and diagnosis are used in research and industry and the choice of an approach is driven by the complexity of the system. Broadly, FDD methods can be divided into three categories: Quantitative, Qualitative and Process history-based [12]. The first two categories together constitute model-based diagnosis. The model is usually founded on some fundamental understanding of the system dynamics. This diagnostic scheme needs accurate process models, semi-quantitative models, or qualitative models. On the other hand, the third category comprising of process-history based methods does not assume any form of model information and relies only on past information [8]. Each method designs a measure such as a residual, statistic, energy level or filtered signal to be used for detection. These features are then evaluated using relevant diagnostic measures [6].

Quantitative methods express the physical understanding of the system in terms of quantitative information related to the inputs, outputs and normal behavior of the system. These quantitative models work on generating and analyzing inconsistencies between the actual and expected behavior [13]. These inconsistencies, called residuals, are larger in the presence of a fault, thereby leading to fault detection. The residuals are obtained by installing multiple sensors and using actual measurements of inputs and outputs or by

devising accurate physics-based models derived from the differential equations governing the behavior of the system [8]. The latter is a more popular technique owing to lower costs and greater feasibility. Models based on physical characteristics of the system will be dealt with in greater detail later in this chapter.

Qualitative methods involve designing models based on a knowledge base, heuristic judgment and adaptive learning. At core, they involve designing an expert system which mimics human decision making based on a plethora of if-then-else rules. However, these conditional or logical filters increase manifold as the systems get more complex. In such cases, an algorithm known as abduction reasoning is employed, which involves proposing a hypothesis for the cause of abnormality based on what has been observed. A major downside of these models is a lack of physical understanding of the system which then mandates a continual update of the knowledge base [14].

On the opposite end of the spectrum of FDD methods lie process history-based techniques. These strategies only require easy access to large quantities of information-rich archival data. There are established ways in which this data can be transformed to more meaningful forms and presented as *a priori* knowledge to the diagnostic scheme. This process is called feature extraction [8], [15]. Different statistical and non-statistical methods are used to extract and evaluate these features from the data, to characterize a healthy or a faulty condition.

Such a broad division of FDD methods - into Quantitative, Qualitative, and Process history-based methods – encompasses innumerable techniques and algorithms to achieve fault detection and diagnosis. However, it is useful to study a narrower, more focused classification of FDD methods to gain detailed knowledge of the following selected few. Among the entire range of processes to achieve characterization of healthy behavior and fault detection, the following methods stand out, on account of their robustness, reliability and popularity:

- Physics-based models
- Data-based classification models
    - Neural Networks (NNs or Nets)
    - Support Vector Machines (SVMs)
    - Sparse Linear Discriminant Analysis (SLDA)

2.2.1 Physics-Based Models

A physics-based model is a mathematical representation of a system's behavior that incorporates process knowledge and dynamic entities such as forces, torques and energies into the model, permitting cross-functional analysis of a system's performance [16]. Given an accurate model, fault detection and diagnosis can be performed over a wider operating range with better fault isolation and is computationally less intensive [13].

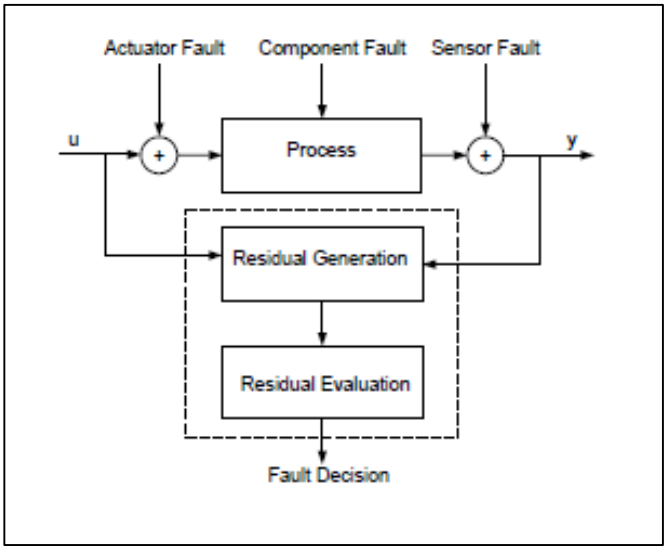

Figure 2.1: Residual generation and evaluation through analytical redundancy – *courtesy [13].*

A typical organization for a physics-based fault detection and diagnosis strategy is shown in Figure 2.1. Two key stages of this method are residual generation and residual evaluation [17]. A residual is essentially an irregularity observed between the modeled and actual behavior of the system. A residual generation is pursued by designing a redundancy around a subsystem of interest. This can be achieved in two ways: physical and analytical approaches [12]. Physical redundancy involves using cluster of hardware to measure or monitor the behavior of a particular component. A malfunction is flagged when one sensor deviates from its nominal behavior. A component failure is registered when all the units deviate drastically from normal [8]. These sensitive instruments require very careful handling, take up more space, cost more to set up, and the overall instrumentation is very high. Hence this is not the preferred method of the two.

The initial introduction of a physics-based model made previously is motivated by the residual generation using analytical redundancy methods. Analytical redundancy methods compare the actual system behavior against the model behavior for residuals. This is achieved from the functional dependence of the process variables and algebraic equations of states and input-output relationships. Various approaches for fault detection using such analytical redundancy-based mathematical models have been developed in the last two decades. Pilot works of such kind were pioneered by [18] and [19]. Willsky surveyed a variety of techniques ranging from specific failure sensitive filters to development of jump process formulation [18]. Himmelblau described usage of tools like process control charts and flow graphs along with state and parameter estimation [19]. The ultimate objective of analytical residual generation is achieved using parameter estimation [20],[11],[21] state observers [22],[23] and signal models [24],[25]. In practical cases, process parameters are not known exactly. Then parameter estimation techniques can be employed using accurate process models; the estimation is carried out by measuring input and output signals [26]. State observers are used for temporal redundancy generation when dynamic relations exist between variables [13]. Signal models are used to ascertain faulty conditions in the presence of oscillatory deviations in the harmonic or stochastic nature of the measured signals. A common principle for residual generation is parity equations. Such an approach represents the consistency relations between inputs and outputs in the discrete domain [13].

Residual evaluation follows whichever of the above methods is used. Residual vectors usually contain unwanted noise and modelling errors. To accurately conclude the presence of a faulty condition, it is essential to remove these features before declaring a non-zero

residual. This can be addressed through statistical testing by making use of the fact that noise is random while a residual is deterministic or semi-deterministic [17]. Tests like direct parallel testing and multivariate testing can be put to effective use. Expert systems or a fuzzy rule base can also be used to perform residual evaluation.

Fault diagnosis follows fault detection. Diagnosis involves both fault detection and isolation. It incorporates finding the fault location and determining relevant corrective measures. Robustness of a fault diagnosis system is crucial so that the system withstands model uncertainties, noise and disturbances and still is able to detect and isolate faults [13]. Various methods can be used to integrate fault detection and isolation (FDI) into model-based diagnosis. Some of the landmark works in the recent history of model-based FDI methods are enumerated below. Paolella and Cho [26] developed a non-linear scheme based on extended Kalman filters to diagnose faults in engine speed, transmission and wheel speed sensors. Gertler *et al* [27] used multiple structured non-linear parity equations supplemented with low-pass filtering of the residuals to diagnose faults in the intake assembly. Recently, Dutka *et al* [27] proposed dedicated observers to generate residuals and isolate faults by establishing residual thresholds and using special “fault signature charts” [28] to isolate the malfunction. Thus, physics-based methods, as summarized in Figure 2.2, form an important branch of FDD methods.

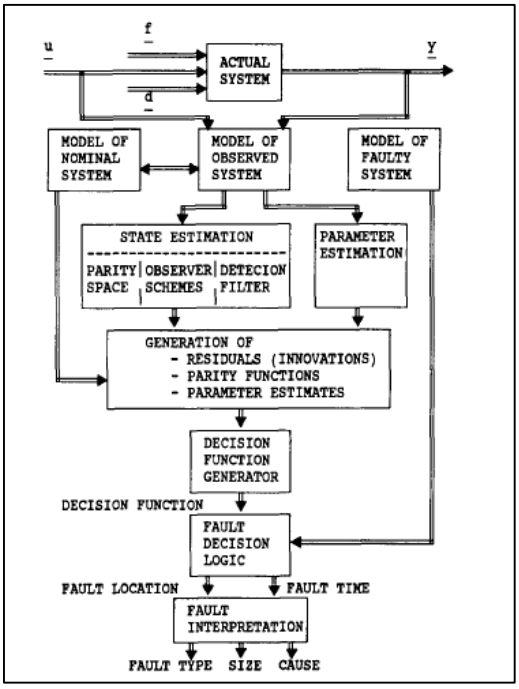

Figure 2.2: Flow chart of fault detection and diagnosis techniques – *courtesy [23]*.

2.2.2 Data-Based Methods

As introduced in Section 2.1, data-based methods do not require a deep understanding of the underlying physics or cross-functional relationships between various parameters. This approach, however, needs a considerable amount of information-rich data. Data-based methods, on account of their simplicity and robustness, have become popular and an important sub-division of FDD techniques.

Within the framework of non-statistical process-history based models, Neural Networks (NNs) are widely used in applications ranging from data processing to computer control to robotics. Inspired from biological neural networks, NNs use dense interconnections of simple computational elements to model complex relationships between inputs and outputs and find patterns in data. A neural net model is specified by net design, node logic and

training or learning modules. The computational elements or "nodes" are adaptive in nature, i.e., they change their logical structure during the learning phase. Learning means developing a generic decision rule based on trends and patterns obtained from the optimal solution of a known set of observations called training data. Data classification problems are well addressed by neural networks. Three-layered feed forward nets [29] provide a straightforward solution to obtaining accurate decision regions required by any data classification algorithms. Therefore, neural networks are widely used for data characterization, fault detection through classification, pattern recognition, etc. However the multi-layer model, owing to its complexity, has high computation and storage requirements.

Support Vector Machines (SVM) originated recently and since then have become a useful tool for binary classification. SVMs and SLDA are closely related in their idea of a decision surface or a hyperplane (a generalized plane in n-dimensions) to achieve classification, regression, etc. The purpose of the decision surface is to separate distinct categories of data by a clear gap, as wide as possible. Support vectors are the points that lie closest to the decision surface [30]. These are most difficult to classify and have a direct bearing on the optimum location of the decision surface. The main aim of a SVM model is to maximize the margin around this decision surface. References [31] and [32] introduce the relevant mathematics of deriving an optimal hyperplane in much greater detail. One of the major advantages of this method is when data is highly non-linear and cannot be separated linearly, SVMs introduce the notion of kernel induced feature space which projects the data

into higher dimensions where it is separable [30]. However, SVMs are more suited to a binary classification where there is only one or the other category for data allocation.

Sparse Linear Discriminant Analysis (SLDA), proposed in [9] is an adapted version of linear discriminant analysis [33]. It addresses the problem of multi-categorical classification of specifying an individual as belonging to one of the many populations to which it possibly can [34]. An analogous multi-fault detection and diagnosis problem can be easily set up using SLDA. In large datasets such as ours, sparseness (low number of non-zero parameters) is desired. SLDA provides this without compromising on the simplicity and robustness of linear discriminant analysis. It achieves input subset selection, dimension reduction and classification in a single step [8],[9]. SLDA finds great use in biology, economics and has been successfully used to classify faulty data in diesel engines through some of the previous works at Herrick Labs at Purdue University [8], [10].

2.2.3 Physics-Based vs. Data-Based Models

A physics-based model design is governed by the underlying working of a system. The first and foremost requisite is the prior knowledge and understanding of the system dynamics. A standard, healthy behavior of the system is built into the differential equations, state-space models that are derived from the static and dynamic nature of the system, and these models are validated and ‘tuned’ using the experimental data. Since these are derived from the intrinsic nature of the system, accurate models can accurately answer the questions related to the causality of faulty behavior by effectively feeding the residuals back to pinpoint the nature of any anomaly.

Data-based models, on the other hand, find their starting point in the data itself. This data is obtained through experiments, in our case through field tests. Model structure is selected based on the nature of data and the parameters are also determined using the data.

Data-based techniques are more advantageous as compared to physics-based models when the focus is quick detection of faulty conditions and not as much finding their cause. Even without an accurate physical understanding, data-based methods work fine while diagnosing faulty conditions. This is not possible in physics-based models that require a clear knowledge of the system, which isn't always possible with modern complex machines [8].

Within the various data-based diagnostic methods that are available and the ones discussed, SLDA is preferred in this work as a potential solution strategy to the previously defined problem of characterization of healthy data and classifying the faulty, flagged data in the outlying clusters. SLDA furnishes linear boundaries and hence makes for a more stable classifier. Linearity also has an added advantage of achieving a greater performance speed and efficiency. There are no expensive computation platforms required to carry out SLDA. More important ideas of discriminant analysis and SLDA will be expanded further in the subsequent chapters.

CHAPTER 3. DISCRIMINANT ANALYSIS FUNDAMENTALS AND SLDA

3.1 Discrimination and Classification

The task of creating a distinction between two sets of data can be accomplished using discriminant analysis where a discriminant function uses a linear combination of the observed parameters that is used as a rule for population separation. Using this knowledge of separating the populations, the problem of classification can be addressed. Classification implies allocating an individual entity to one of the many groups to which it can possibly belong. Best classification would lead to the smallest error rate of all future allocations or be the one with the smallest probability of misclassification [35].

There is thus a small but significant difference between discrimination and classification. A discriminant function aims to maximize the separation between available groups while classification rules minimize the misclassifications over all possible allocations [35]. Many allocation rules are possible, and some of these are described in this section.

Let us suppose x denotes an n-dimensional random vector of observations made on any occurrence. The vector-valued observations x_1, x_2 ...etc. denote specific observed values of x. Let φ_1, φ_2 denote the two populations under study. (We take a case of binary classification for explanatory purpose. In application, there may be more categories than

two). To distinguish the two populations, the fundamental assumption is that x has different probability distributions p_1 and p_2 in φ_1 and φ_2, respectively. Simplest of the allocation criteria with above structure is the likelihood ratio:

$$Allocate\ x\ to\ \varphi_1\ if\ \ p_1(x)/p_2(x) > 1\ \ and \quad (3.1)$$
$$to\ \varphi_2\ if\ \ p_1(x)/p_2(x) \leq 1.$$

However the likelihood ratio is often too simple a criterion for classification. It suffers due to the absence of two vital factors – prior probability and cost of misclassification. The prior probability distribution or just prior is the probability distribution representing knowledge or belief of occurrence of an event prior to collecting or observing any data. The term 'cost of misclassification' is self-explanatory. It is the excess cost incurred for wrongly classifying the data. The word 'cost' can have a broader, subjective sense than just monetary loss. It implies the damages or forfeitures of varying natures and degrees which can be caused by misallocations of new data. From an engine manufacturer's perspective minimizing false positives is therefore very important [3], [8]. Erroneous flagging of engine states as faults may lead to unnecessary warranty claims and partial or total replacements which amount to tens of thousands of dollars. Hence, it is vital to take these two factors into account while devising an allocation rule. Following is one such criterion which accounts for priors and cost of misclassification:

$$Allocate\ x\ to\ \varphi_1\ if\ \ p_1(x)/p_2(x) > c(1|2)\gamma_2/c(2|1)\gamma_1\ \ and \quad (3.2)$$

$$to\ \varphi_2\ if\ \ p_1(x)/p_2(x) \leq c(1|2)\gamma_2/c(2|1)\gamma_1,$$

where $c(1|2)$ is the cost of misclassifying an individual which actually belongs to φ_2, to φ_1. Similarly, $c(2|1)$ carries the opposite sense. The values γ_1 and γ_2 are the prior probabilities of belonging to φ_1 and φ_2, respectively.

Posterior probabilities of an event can be derived using the quantities above; *a posteriori* carries an exactly dual meaning to *a priori.* It is the probability computed after taking the occurrence or relevant data into account [33]. Using Bayes' theorem, a posterior probability - that an observation x_1 comes from φ_1 – can be expressed as

$$\mathrm{P}(\varphi_1|x_1) = \gamma_1 p_1(x_1)/\big(\big(\gamma_1 p_1(x_1) + \gamma_2 p_2(x_1)\big). \tag{3.3}$$

Then a workable criterion for classification of x is to

$$allocate\ x\ to\ \varphi_1\ if\ \mathrm{P}(\varphi_1|x) > \mathrm{P}(\varphi_2|x)\ and \tag{3.4}$$
$$to\ \varphi_2\ if\ \mathrm{P}(\varphi_1|x) \leq \mathrm{P}(\varphi_2|x).$$

These three rules are among the many corollaries of 3.1 and 3.2. However, all these criteria are based on the knowledge of the probability distribution functions, p_1 and p_2, of the two populations. In most practical cases, the parameters of these probability models are not known directly. In fact, the only meaningful data available is the first set of observations and their respective class labels. This data, called training data, is used to estimate the parameters of distribution functions and thus directly influence the classifying criterion [35],[33].

Suppose we assume a multivariate Gaussian model of the following form with a mean μ and a covariance matrix Σ:

$$p_c(x) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_c|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_c)^T \Sigma_c{}^{-1}(x-\mu_c)} \tag{3.5}$$

where n indicates the dimensions of the input space, i.e., the number of parameters every observation is drawn upon, and the subscript c denotes the class marker. The mean μ_c and covariance matrix Σ_c are estimated using the training data as follows:

$$\hat{\mu}_c = \frac{\sum_{i=1}^{N_c} x_i}{N_c} \tag{3.6a}$$

$$\hat{\Sigma}_c = \frac{1}{N_c - 1} \sum_{i=1}^{N_c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T. \tag{3.6b}$$

where N_c is the number of observations in class 'c' training data.

From equations 3.1 to 3.6, a well-rounded picture of binary classification is obtained. However, this set of equations is rarely used as is to address discrimination and classification owing to their complexity and computational requirements. Moreover, a sizable simplification can be achieved by assuming equality of the two dispersion matrices in the problem, with $\hat{\Sigma}_1 = \hat{\Sigma}_2 = \hat{\Sigma}$. Now,

$$\frac{p_1(x)}{p_2(x)} = \exp\left\{(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1} x - \frac{1}{2} (\hat{\mu}_1 - \hat{\mu}_2)^T \Sigma^{-1} (\hat{\mu}_1 + \hat{\mu}_2)\right\}. \tag{3.7}$$

From equation 3.2, taking the R.H.S. - $c(1|2)\gamma_2 / c(2|1)\gamma_1$ – as a constant k, 3.2 is transformed to

$$\begin{aligned} &Allocate\ x\ to\ \varphi_1 \\ &if \exp\left\{(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1} x - \frac{1}{2} (\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1} (\hat{\mu}_1 + \hat{\mu}_2)\right\} > k\ and \end{aligned} \tag{3.8}$$

$$to\ \varphi_2\ if\ \exp\left\{(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1} x - \frac{1}{2}(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1}(\hat{\mu}_1 + \hat{\mu}_2)\right\} \leq k.$$

A logarithmic transformation to both sides of 3.8 yields,

$$Allocate\ x\ to\ \varphi_1\ if\ \delta(x) > log_e k\ and\ otherwise\ to\ \varphi_2$$

where $\delta(x) = (\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1}\left\{x - \frac{1}{2}(\hat{\mu}_1 + \hat{\mu}_2)\right\}$.

The function $\delta(x)$ is a discriminant function and since there are neither normalization nor quadratic factors involved in the expression, it is called *a linear discriminant function.* This implies that the decision boundary – that part of the n-dimensional space where an observation with *n* parameters bears equal probabilities of belonging to either population – is linear in *x*: a hyperplane in *p*-dimensions. In fact, $(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1}$ is a row vector, say $\alpha = (\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_p)$. Thus if $x_1, x_2, x_3 \ldots x_p$ represent the elements of an observation x, $\delta(x) = \alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots\ldots\ldots + \alpha_p x_p$, where $\alpha_0 = -\frac{1}{2}(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1}(\hat{\mu}_1 + \hat{\mu}_2)$ and $x_0 = (1,1,1 \ldots 1)^T$

3.2 Fisher's LDA

Until now, the discriminant functions and classification rules were based on assumptions of parametric forms of the probability models. In contrast to this, Fisher proposed a discriminant analysis method from a purely data-based standpoint, without assuming the shapes of underlying distributions, if any. Fisher's linear discriminant function is an effective way to obtain a lower dimensional representation of the data while also utilizing the class label information of these projections [36]. To this end, Fisher's LDA aims to maximize

$$J(\alpha) = \frac{\alpha^T S_b \alpha}{\alpha^T S_w \alpha} \tag{3.9}$$

where S_b and S_w are "between-class variance" and "within-class variance", respectively, and α is a set of coefficients that comes from $(\hat{\mu}_1 - \hat{\mu}_2)^T \hat{\Sigma}^{-1}$, as described previously. Here,

$$S_b = \sum_c (\hat{\mu}_c - \bar{\mu})(\hat{\mu}_c - \bar{\mu})^T \tag{3.10a}$$

where $\hat{\mu}_c$ is the class "c" mean and $\bar{\mu}$ is the overall data mean. Similarly,

$$S_w = \sum_c \sum_{i \epsilon c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T \tag{3.10b}$$

where the x_is are individual elements of class c and $\hat{\mu}_c$ is the class "c" mean. Fulfilling this objective intuitively makes sense too. It suggests that class means are well-separated, and data belonging to a class are clustered together effectively. This is exactly what is needed to satisfy the objectives of characterizing the healthy data cluster and separating it from the smaller clouds of data representing anomalous behavior.

For a binary classification problem (two classes), it can be shown that the vector α that maximizes J should satisfy

$$S_b \alpha = \lambda S_w \alpha, \tag{3.11}$$

which can be written as

$$S_w{}^{-1} S_b \alpha = \lambda \alpha, \tag{3.12}$$

which is a generalized eigenvalue problem [36]. Solving for the eigenvalues and eigenvectors of this problem is not necessary as $S_b \alpha$ lies in the direction of $\mu_1 - \mu_2$. Hence,

$$S_b\alpha = \kappa(\mu_1 - \mu_2), \tag{3.13}$$

i.e.,

$$S_b\left\{\frac{1}{\kappa}.\alpha\right\} = (\mu_1 - \mu_2). \tag{3.14}$$

From the expression for J (3.9), rescaling of α is immaterial. Hence, taking $\kappa = 1$, $\alpha = {S_b}^{-1}(\mu_1 - \mu_2)$ presents an acceptable solution to the binary classification cum LDA problem.

3.3 Sparse Linear Discriminant Analysis

For many cases, linear discriminant analysis works well. However, there are many practical applications where LDA is not the best strategy. These are the problems of discriminant analysis that involve a large number of predictor variables. In such cases, feature selection or identifying the parameters that best characterize the Fisher's criterion is desired since it promotes faster and more accurate classification. With large number of predictors, another issue is maintaining the non-singularity of the resultant large dispersion matrices. Hence, sparseness is also desired along with feature selection. Sparseness ensures a non-zero loading of the dispersion matrix. Sparse Linear Discriminant Analysis (SLDA) induces an effective sparseness criterion such that it performs discriminant analysis, feature selection and classification all in one step [9]. This helps provide a better interpretation of complex data and is less likely to overfit the training data compared to other methods. SLDA also provides the very informative low-dimensional projections of data which help visualize the separation of "*K*" classes as clusters in "*K-1*" dimensions.

SLDA uses the Fisher's criterion

$$J(\alpha) = \frac{\alpha^T S_b \alpha}{\alpha^T S_w \alpha}, \tag{3.15}$$

where S_b and S_w are between-class and within-class variances and α_js represent the "*K-1*" directions. The most suitable α_js are those that maximize the variance between classes w.r.t. variance within the classes and are orthogonal to each other. Mathematically, the same can be written as solving for

$$\arg \underbrace{max}_{\alpha_j} \alpha^T S_b \alpha. \tag{3.16}$$

Previously, Hastie *et al* (1995) introduced a penalty $\lambda_2\Omega$ on the within-class variance to impose a spatial smoothness constraint on the coefficients such that $S_w \rightarrow S_w + \lambda_2\Omega$ [37], [9]. Here, Ω is a non-zero, symmetric penalty matrix. In SLDA, an extra factor is introduced to regularize the l_1 norm of the coefficients to induce sparseness in the dispersion matrix. The discriminant criterion with sparseness becomes

$$\arg \underbrace{max}_{\alpha_j} \alpha^T S_b \alpha - \lambda_1 \sum |\alpha_{ji}|, \tag{3.17}$$

under the constraint $S_{wp} = S_w + \lambda_2\Omega$ as per the penalized analysis of within-class variance.

A small digression to review regression helps complete the picture of SLDA. In the usual regression problem to predict a numerical outcome, $\hat{y}$ is built on arrays of predictor variables, x_is and regression coefficients, α_is. This can be expressed as

$$\hat{y} = \alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots\ldots\ldots + \alpha_n x_n. \tag{3.18}$$

Various methods are commonplace to ascertain the existing unknown parameters of this expression while introducing regularization functions to improve predictability of the model. The simplest method uses ordinary least squares (OLS), which estimates the model

variables by minimizing the residual sum of squares (RSS) [38]. However it is established that standard OLS does poorly both in terms of prediction and interpretation in many large-dimensional real-time applications. Penalization techniques have since come to the fore to improve OLS. Ridge regression puts a bound on the L_2 norm of the coefficients to minimize RSS [39], [40]. However, scientific studies demand a model with minimal number of predictor variables to put more light on significance of relationships between the covariates and the outcome. Ridge regression fails here since it uses all the available predictors, leading to a less interpretable model. To this end, Tibishrani [41] proposed LASSO or Least Absolute Shrinkage and Selection Operator method. LASSO imposes a penalty on the L_1 norm of the coefficients, which ensures automatic feature selection. However, LASSO fails in two commonly occurring cases:

i. Number of predictors (p) is greater than number of observations (n) – In cases of $p > n$, LASSO exhausts while selecting only n predictors. This is a limiting feature for a generalizable variable selection method.

ii. High correlations within predictors – If a high degree of pairwise or grouped collinearity exists, LASSO tends to select only one of the many related variables at random.

To account for these shortcomings, a novel method called “elastic net regularization” was proposed by Zou and Hastie in 2005 [40]. Elastic Net betters the LASSO by imposing a second penalizing term on the L_2 norm which by itself is used in Ridge regression. Elastic net also achieves continuous shrinkage and automatic subset selection of input predictors.

It has the capability of selecting or rejecting a group of correlated variables, which leads to better interpretation of the model.

The regularization condition for Elastic net is

$$\alpha_j{}^* = \arg\underbrace{min}_{\alpha}\left(\left|\left|y - X\alpha\right|\right|^2 + \lambda_1\left|\left|\alpha\right|\right|_1 + \lambda_2\left|\left|\alpha\right|\right|_2{}^2\right), \tag{3.19}$$

where

$$||\alpha||_1 = \sum_{j=1}^{n} |\alpha_j|$$

$$||\alpha||_2 = \sum_{j=1}^{n} \alpha_j{}^2.$$

The elastic net, by using two penalty terms, retains the properties of a LASSO ($\lambda_1 = 1, \lambda_2 = 0$) and Ridge regression ($\lambda_1 = 0, \lambda_2 = 1$). [10] This is analogous to the SLDA criterion in 3.17 which also uses two penalizing terms for better assurance of sparseness. Hence, it is advantageous to rewrite the discriminant condition as a regression problem based on elastic net regularization.

3.3.1 Classification based on SLDA

The outputs of a regression problem are numeric or quantitative in nature, whereas a classification problem has categorical results. This concern over the categorical nature of the output of a classification problem and its conversion to quantitative forms is addressed using the optimal scoring technique proposed by Hastie *et al* [38]. Optimal scoring assigns scores to each class based on certain predetermined rules and converts the categorical allocations to numeric forms [9], [8]. An iterative algorithm proposed in [9] is used to solve this regression problem achieved by optimal scoring. The quantitative outputs of this

process are converted back to categorical variables for class allocations of the data. The selection of the values of λ_1 and λ_2 is governed by the nature of the data so that these parameters optimize the overall process of classification to the one with least error rates.

Until now, we have spoken about the design of a decision rule with imposed sparseness within the premises of SLDA. This is done using the already-existing training data with known class labels. The solution of the regression problem based on the SLDA decision rule achieved the classification of the given data. The picture gets completed with the satisfactory evaluation of the SLDA classifier with a completely new, yet-unseen validation set or the test data. Testing of a classifier will be covered in Chapter 4.

CHAPTER 4. SLDA IMPLEMENTATION: INDIVIDUAL TRUCK ANALYSIS

The theory of linear discriminant analysis was introduced in the previous chapter and the statistical improvements that led to the development of Sparse LDA were briefly highlighted. In this chapter, SLDA is described from a more practical perspective where it is applied to actual data from field-tested trucks. There are quite a few intermediate data processing steps which improve the quality of overall analysis and the accuracy of classification as well. These steps will be laid out in Sections 4.2 and 4.3. The MATLAB code for SLDA is then explained from a "blackbox" perspective where the inputs and outputs are studied in detail. Finally, the graphical and numerical outcomes for individual trucks are discussed.

4.1 Engine Data

At the outset, it is useful to shed some light on the nature of the data used in this work. All data was obtained through field tests of 15 *l* heavy-duty diesel engines manufactured by Cummins Inc. All the engines were used as on-road transport applications. The truck code names with power and torque ratings are summarized in Table 4.1.

Table 4.1: Field test truck power-torque pedigree chart.

Truck	Power (hp)	Torque (lb-ft)
103	450	1750
8572	600	2050
231	500	1650
232	500	1650
234	500	1650

As seen from the ratings, trucks 231, 232 and 234 belong to the same engine family, having similar ratings and calibrations. Moreover, they had identical failure modes in our analysis. The failure mode in these trucks - collectively called the T200 series - was revealed to be an EGR leak in the crossover tube between the EGR measurement orifice and the intake system. In addition, data was collected under similar ambient and road conditions in an eighty mile periphery around Portland, Oregon [10].

T103 and T8572 contain engines designed for different ratings, with distinct hardware and calibrations. Their failure modes were unknown and different from one another and also from the T200 series. The data was collected over the months of May to October, which covers a broad region of ambient conditions, duty cycles and geographic locations. This offered a challenging opportunity to come up with pre-processing techniques and modifications to the base program for implementing SLDA to account for these differences in data and still find common ground to analyze the trucks individually and in groups of two or more (Chapters 5 & 6).

4.1.1 Data Acquisition

It is useful to bear in mind the ways in which the data was collected. For each truck listed in Table 4.1, one healthy and one faulty data set was collected. Each data set was a day's worth of engine operation, acquired at a rate of 1 Hz. Care was taken to use the data only from those days when the truck/machine operation was sufficiently high - generally more than fifteen hours. First, the faulty data was collected immediately after the operator noted a lighted MIL – "Check Engine" lamp. The engine was then taken to the service department for diagnosis and remedial measures. After service, the first set of a full day's data obtained was used as the healthy set. Generally, the healthy set was collected a few weeks after the collection of faulty data.

4.2 Signal & Data Selection

SLDA's feature extractor, when run in an iterative loop, is analogous to a stepwise feature selection in regression analysis, i.e., for every 'm' iterations it chooses a subset of 'm' variables that best expresses the separation between healthy and unhealthy data. In the previous chapter, we have seen the expression for discriminant function, $\delta(x) = \alpha_0 x_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots\ldots\ldots + \alpha_n x_n$, (where $x_1 \ldots x_n$ are variables that make up an observation) and thereby the importance of selecting a universal space of key variables. For our work, the idea was to encompass multiple subsystems of the engines. Focus was on standard performance variables like speed, torque, fueling and other sensor inputs pertaining to EGR (Exhaust Gas Recirculation), oil, and exhaust, etc. Measurements of ambient conditions like air pressure and temperature were also included. Aftertreatment signals related to DOC (Diesel Oxidation Catalyst), DPF (Diesel Particulate Filter) and SCR (Selective Catalytic

Reduction) were ignored since the scope of work demanded focus on engine performance. After much deliberation with engineers at Cummins, twenty-six variables out of more than 450, stored by a standard data logger, were finalized to be used in the model. Based on every truck's behavior, the algorithm selects the best subset of signals for classification.

Similarly, not all of the raw data available was used as input to SLDA. It was observed that most trucks, while in highway operation, run in certain ranges of speed and load for a majority of time. After a few preliminary iterations of the algorithm, it was observed that the data pertaining to highway operations yielded better results. This data was obtained by using specific filters devised in consultation with Cummins. The filters used are listed as below:

- $Engine\_Speed \geq 1000\ [RPM]$
- $Net\_Engine\_Torque \geq 0\ [ft-lb]$
- $Total\_Fueling \geq 50\ [mg/strk]$
- $Coolant\_Temperature \geq 71\ [deg\_c]$

Using these filters actually improved the overall classification [10] for individual truck analysis. On a side note, prior to these filter applications, all the NaN (Not-a-Number) values were removed.

4.3 Data Matching

Since the data was acquired in a non-experimental setting, the problem of selection bias must be addressed [42]. In the problem of causal inference, this selection bias tends to confound the classifier and increases the number of misclassifications. Section 4.1.1 clearly

indicates that there is a significant time lapse between collection of faulty and healthy data. Between acquiring the two kinds of data, external factors like ambient conditions, test routes, drivers and their operating choices may change drastically. These factors directly affect the intrinsic nature of data. For example, data collected on a hot day bears natural differences from the data collected on a cold day. These differences between the healthy and faulty, which may not necessarily correspond to presence of a fault, tend to confound the classifier. Figure 4.1 shows histograms of Compressor Inlet Temperature (CIT) data for healthy and faulty datasets of Truck 8572. Changes in CIT are a direct result of variations in the ambient air temperatures. Notice the extent of dissimilarity between the two histograms. Hence an algorithm to compensate for these differences was deemed necessary.

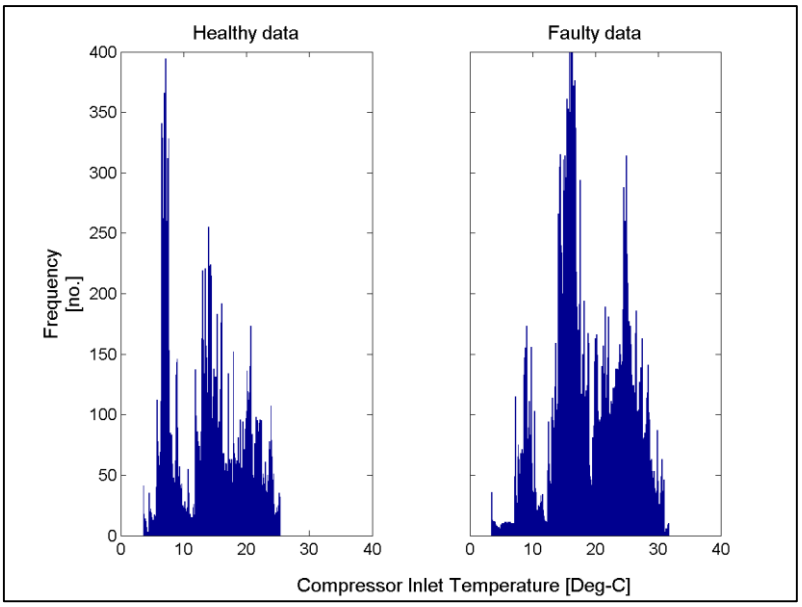

Figure 4.1: Differences in healthy – faulty manifesting importance of matching.

We have developed an exploratory algorithm of rank-based matching to achieve this. At the outset, it is essential to know that the matching rule is based on a selected few signals only. These five signals are speed, fueling, air flow, air pressure and compressor inlet temperature. Operating conditions are captured using speed and fueling and the environmental conditions are characterized by the other three. The idea here is to remove the differences between healthy and faulty within these signals since they may not necessarily indicate a faulty condition. The algorithm works in the following way:

- First step of data processing is the removal of NaNs. This is followed by applying data selection filters and calculating statistical features such as moving means, moving std. deviations, moving differential means etc. A 120 sec moving window was used for this based on the results obtained in [10].
- This is done for both healthy & faulty data.
- A faulty point is taken in order of time and the differences of its four abovementioned statistical quantities with those of all the healthy points are calculated.
- These differences are sorted and ranked, and the healthy point with the least difference (or the top rank) is chosen as a pair to that one faulty window.
    - Typically, more than one healthy window could be chosen for multiple faulty windows because it provided the best match!

This process is illustrated in the following figures. In Figure 4.2, XF represents faulty data and XH represents healthy data. Consider a case of one variable, with 10 windows of points and four moving statistical quantities each – A, B, C, D for faulty and a, b, c, d for healthy.

Specifically, these are moving means, moving standard deviations, moving differential means, and moving differential standard deviations. Differences are found for a moving quantity of a faulty window with the corresponding moving quantities of all the healthy windows (Figure 4.3). These differences are ranked and summed as shown. The best ranked window with smallest sum of differences of healthy data is chosen corresponding to this one window of faulty data. Here it is xh8.

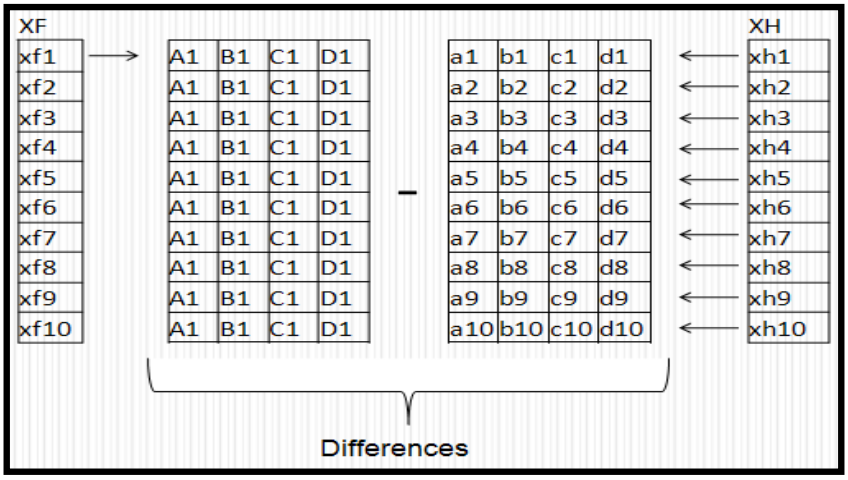

Figure 4.2: Rank-based matching - differences between statistical features corresponding to one faulty window and all the healthy windows. – *courtesy [10]*.

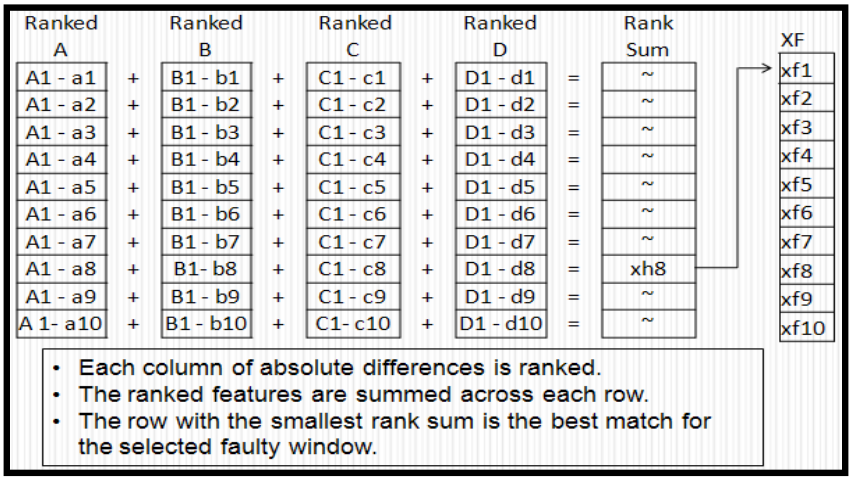

Figure 4.3: Rank based matching - ranking the differences and selecting the highest ranked window as the best match.

In this work, rank-based matching was based upon four features obtained via healthy-faulty differences of moving means, moving standard deviations, difference of successive moving means, and difference of successive moving standard deviations. Five key signals that best express the operating and environmental conditions during the day were chosen for the above-mentioned calculations. These were engine speed, total fueling, fresh air flow (operating conditions), compressor inlet temperature and ambient air pressure (environmental conditions). The four features described were calculated for each of the three operating conditions, while only moving means and moving standard deviations were estimated for the environmental conditions. This is because they usually do not change considerably in a day. Thus, in this work, sixteen features were used for the rank-based matching. Since the ranks of differences are added, the variations in dimensions do not need to be accounted for. The ability to select the same healthy window multiple times ensures consistently good matching.

Figure 4.4 shows two histograms of differences between healthy and faulty values of CIT for pre- and post-matching data conditioning. The histogram after matching looks more normal with most of its data centered at zero difference, which indicates good matching. The span of the histogram reduces too – this ensures that even at "not-so-good" match, we pick data from reasonably similar regions of operation. This holds for the other signals used in matching. Thus, rank-based matching effectively helps to remove the selection bias.

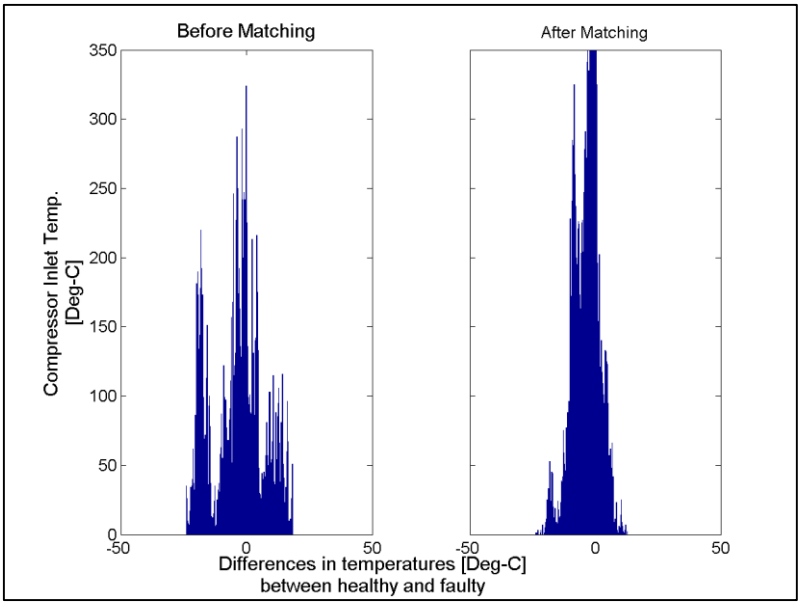

Figure 4.4: Pre- and post-matching snapshots of compressor inlet temperature data.

4.4 SLDA Implementation

A quick review of the problem: Given some random data $X \in \mathcal{R}^n$, known to belong to classes with labels $c_1 \dots c_k$, the task is to classify a new, yet-unseen set, $Y \in \mathcal{R}^n$, believed to be future data associated with X, into classes $c_1 \dots c_k$. Clearly, X and $c_1 \dots c_k$ become two of the inputs to the SLDA code, and thus X and $c_1 \dots c_k$ together make up the training data. In this work, twenty-six measurement signals were considered (refer to section 4.2), out of which five were used for matching the operating and environmental conditions. The remaining twenty-one constitute every observation of data. The smallest dataset belonged to Truck 231 – 20685 points, after applying the previously described data processing. To maintain equivalence and equal data sizes, this same number of points (#1 to #20685) was separated and used from every truck. Out of the available 20685, 95%, i.e. 19650, were used as the training data and 200 points were randomly selected from the remaining 5% for validation. Both training and test data were normalized to account for dimensional

differences in variables. For the first stage, i.e., individual truck analysis, there can exist only two classes – healthy and unhealthy (or faulty data), encoded as "0" and "1", respectively.

The two inputs to SLDA, thus, are the following two matrices:

$$concat_{train} = \begin{bmatrix} {xh_1}^1 & \cdots & {xh_1}^{21} \\ {xh_2}^1 & \ldots & {xh_2}^{21} \\ \vdots & \ddots & \vdots \\ {xh_{19650}}^1 & \cdots & {xh_{19650}}^{21} \\ {xf_1}^1 & \ldots & {xf_1}^{21} \\ {xf_2}^1 & \ldots & {xf_2}^{21} \\ \vdots & \ddots & \vdots \\ {xf_{19650}}^1 & \ldots & {xf_{19650}}^{21} \end{bmatrix}$$

$$dummy_{train} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1_{19650} & 0_{19650} \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0_{19650} & 1_{19650} \end{bmatrix}$$

The matrix $concat_{train}$ has normalized columns; it is a concatenated matrix of healthy (xhs) and faulty (xfs) data of dimension (39300×21). Matrix $dummy_{train}$ is a binary-coded matrix corresponding to the location of healthy and faulty data in $concat_{train}$.

Besides $concat_{train}$ and $dummy_{train}$, there are three other secondary inputs to SLDA, secondary because SLDA will assume default values and still work without these inputs. However, these are vital parameters that help to 'fine tune' the performance of the classifier. Based on certain kinds of data, one of the three may be more important than the others. These are:

1. Lambda: It is λ_2 or the weight on the L_2-norm for the elastic net regularization method [9], as in equation 3.19. It is also called Ridge term coefficient [8] since it controls the influence of ridge regression on the elastic net. Default value is set to 10^{-6}.
2. Stop: A stop value controls various aspects of the SLDA criterion.[43]
    a. Non-zero value of stop enables elastic net algorithm on the data with early stopping.
    b. In the MATLAB implementation of SLDA, if stop is negative, its absolute value indicates the maximum number of parameters that can be used by the feature selector. Referring to the previous section, it was seen that $X \in \mathcal{R}^n$. Here, n is the number of parameters or dimensions defining the observation X. A stop value provides control to keep the number of features below n. In case the stop value is a set or array of numbers, SLDA is performed at every stop. The results will provide a better view of this notion of iterative stop.

c. A positive stop value puts an upper bound on λ_1 or the L_1-norm of equation 3.19. This controls the effect of LASSO solution in elastic net.

3. maxIter: This parameter sets the maximum number of iterations that the algorithm performs at the specified value(s) of stop. The default value is 25; however in this case, the values rarely exceed 3.

Additionally, there are other parameters: *'disp'*, which switches on or off the display of key variables, and *'tol'*, which sets tolerance values for the residual sum of squares, which also acts as the stopping criterion of the algorithm. Default and sufficient tolerance is 10^{-6}.

The SLDA code furnishes three output terms. These are:

1. Alpha or the αs of equation 3.18. These are the coefficients that optimize the Fisher's criterion in equation 3.9,
2. Theta or the optimal scores are necessary to solve the SLDA criterion as a regression problem (Section 3.3),
3. RSS or the residual sum of squares at every *'maxIter'* iteration as described above.

First output, αs, are the sparse directions; these are the directions that maximize the distinction between the two classes and minimize the variance within classes. The projection vector is obtained by multiplying the $concat_{train}$ with $(\alpha_1, \dots, \alpha_{21})^T$. In the case where all of the directions are not exhausted, some of the αs will be zero.

Thus, discrimination is achieved by generating the projection vector. However, another step is necessary to classify these projected points as well as the new, unseen data into two

classes. MATLAB function *classify.m* allocates the data under different labels or groups. To briefly explain, *classify.m* picks every point in the n-dimensional space, calculates its distances from the centroid of two projected clusters of training data and allots the point to the population it is closest to.

A typical expression for the function call of *classify.m* contains five inputs and one basic output – the allotted class:

$$[class] = classify(sample, training, group, 'type', prior)$$

The first three inputs are more fundamental than the last two. Variable $sample$ is the projection vector that needs to be classified. Variable $training$ is the projection data obtained through SLDA. These two inputs must have the same number of columns; this makes sense intuitively because while classifying new data with respect to the training, the number of features used to express the data must be the same. Variable $group$ is analogous to the $dummy$ matrix used in SLDA. It encodes the training data according to the groups it is associated with. The only small difference is that group is an identifier matrix which can contain any two distinct labels with similar or different datatypes. In this work, for a two class problem, healthy is represented by 0, and faulty is denoted by 1. This is not to be confused with the binary encoding of the dummy. The thing to bear in mind is, $dummy$ has to be coded in zeros and ones. The group matrix, on the other hand could have columns like *0* and *1*, or *1* and *2*, or *"Healthy"* and *"Faulty"*, or "*h123"* and *"f456"*! In the upcoming chapters, the difference between group and dummy will be clearer as the number of class labels increases to three or four.

The last two inputs are type of classifier demanded and *a priori* probability of occurrence of entities in each class. By default type, MATLAB uses a linear classifier. Essentially, this means that the boundaries between the classes are linear in nature. Other 'types' can also be specified and are used commonly, but this work is strictly based on linear classification owing to a greater stability of training and lesser issues of overfitting [33]. *A priori* probability (explained in Section 3.1) is based on archival knowledge of the problem and groups of data being considered. The default value is 0.5 for each class, i.e., equal priors. Finally, the output or the solution of the code is the "class" or "group" an observation from the "sample" is allotted to.

4.5 Evaluation Using Test Data

From the beginning of Chapter 3 until now, the process of building a classifier using training data has been addressed. Equally important is unbiased testing of the classifier – "unbiased" in the sense that neither the allocation rules nor the *classify* function should contain the test data while building up the model. Thus, the most common and effective way to generate a validation set is to use only a fraction, albeit closer to 1, of the available data as the training set. Leave-one-out cross validation [44] is one of the most basic derivations of the above rationale. An 80-20 ratio in favor of training with k class cross validation was used previously by the researchers at Purdue [8]. This method arose from the knowledge of necessity of cross validation - just a single iteration with 80% data as the training set and the rest as test might lead to biased results since rates of classifications depend on the intrinsic nature of data. The data being random, it was thought to be common wisdom to use multiple combinations of 80% - 20% within a universal set of data. However

in this work, since classification of multiple trucks is done individually and together in different combinations, it was deemed necessary to use equal lengths of data of each truck to minimize preference and bias towards a specific truck. With unequal number of data points, the priors no longer are unitary as shall be addressed in Chapter 6.

So a training set of 1 to 19650 points was taken sequentially from each truck. The test set comprised of 200 points randomly selected for the remaining data of each truck. Such a selection is more synonymous to the idea of testing against "oncoming, unseen data". This set was separated right after the application of signal and data selection and before matching. Cross-validation was passively performed using a 95% - 5%, training – test division, but not used in the final set of results.

4.6 Preliminary Results

The outcome of SLDA-cum-*classify*.m code in MATLAB is a set of three plots, namely, Error rates, Instances of misclassification, and the ROC curve. Sample plots manifesting important characteristics of these results are included below.

The plot for error rates can be seen in Figure 4.5. As previously mentioned, SLDA and *classify.m* can be made to run on an iterative basis while including an additional variable every time; this is analogous to a stepwise feature selection method. The "error rates" plot shows the change in the error of classification as an extra variable is added on every iteration. Percentage error is plotted on the Y-axis and the index of variable on X. Usually,

in case of training data, the error rates monotonically decrease with addition of a new parameter.

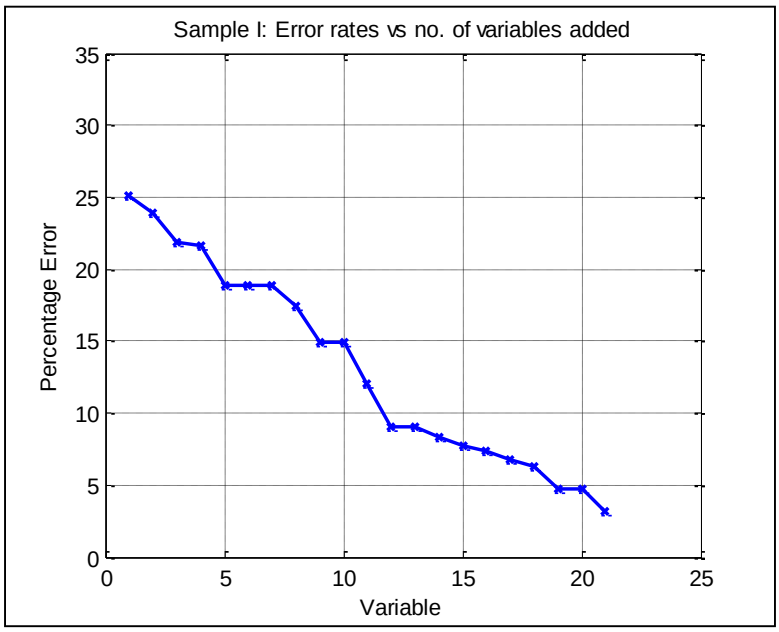

Figure 4.5: Sample plot: error rates vs number of variables.

Figure 4.6 is a sample plot showing the instances of misclassification. While describing *classify.m*, it was mentioned that the healthy and faulty classes are represented by the grouping variables 0 and 1, respectively. This is visualized in Figure 4.6.

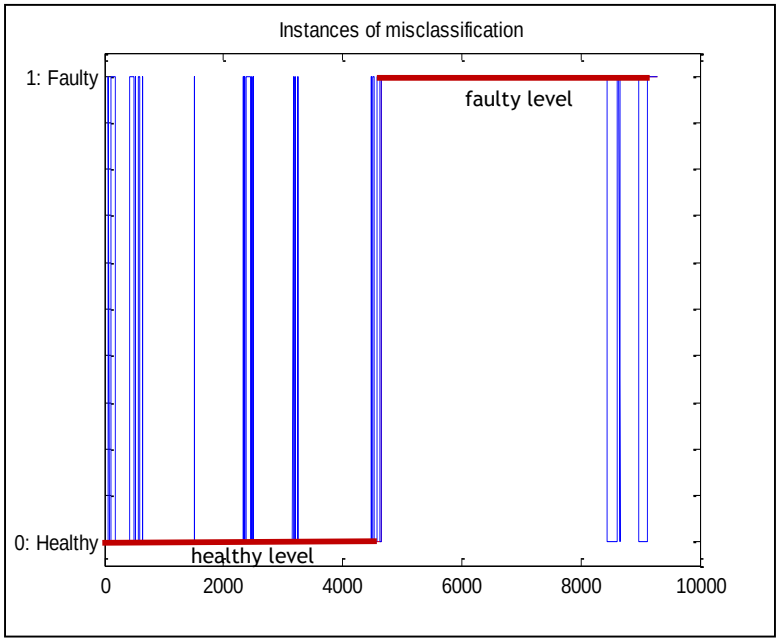

Figure 4.6: Sample plot: instances of misclassification.

Ideally for 100% accuracy, the first half of the data should all lie on the red line at level 0 and second half on level 1. This is because the dummy matrix contains healthy data in the first half and faulty in the other. The bold red segments indicate healthy and faulty zones. The blue segments going up or down away from the red line indicate misclassifications in that zone. For example. healthy data is indexed at about 0 - 4500 points. The blue points are misclassified as faulty.

The final visual is the Receiver Operating Characteristic (ROC) curve (Figure 4.7). The ROC curve illustrates the performance of a binary classifier as its discriminating thresholds are varied. These thresholds are tuned to adjust the rates of false positives. Intuitively, this is essential since besides reducing misclassifications, a manufacturing company would want to reduce the number of false alarms to prevent losses due to unnecessary warranty

claims. Hence, adjusting the parameters so as to keep the false positive rate close to zero is an essential requirement of this work. The ROC curve helps to tune this aspect of classification.

For achieving the objectives of this research, high rates of true positives (good characterization of healthy data) and low rates of false positives (good classification of faulty data) was targeted. Ideally, the most desirable ROC location would be the top-left corner of the plot.

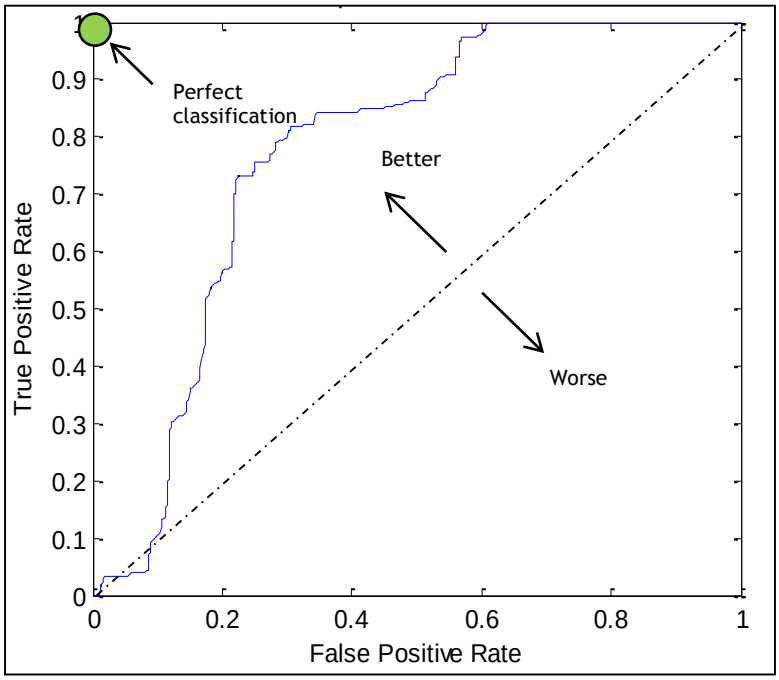

Figure 4.7: Sample plot: receiver operating characteristic (ROC) curve.

4.7 Results: Set I

A note to the reader - there are multiple cases of data setup and further two chapters with multi-truck, multi-fault classification. Hence to check the extent of detail, only the plots of error rates are utilized for comparisons of training and test results as well as the error rates

over different trucks. In this chapter, trucks 103, 231 and 8572 will be validated at an individual level.

The first set of results use basic engine filters on the raw data, matching, and the previously described setup of training and test data. Please follow the titles and labels on the plots for more information.

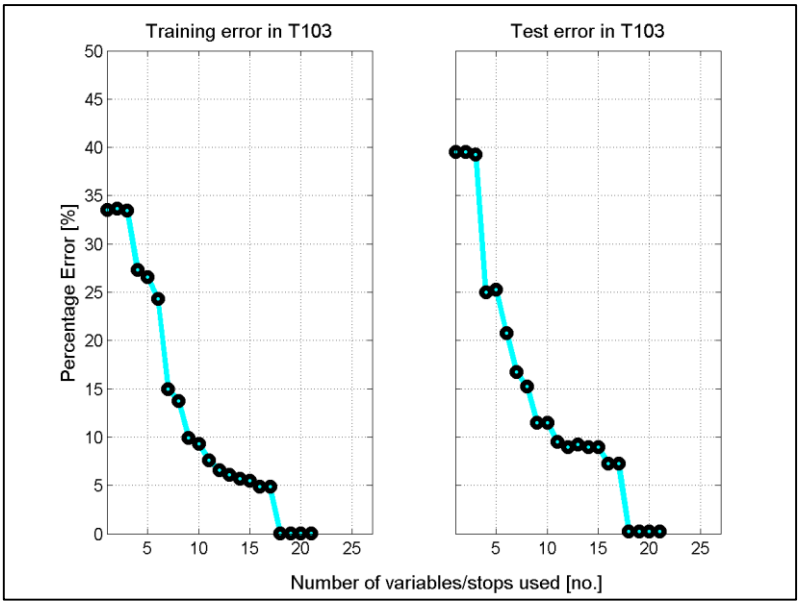

Figure 4.8: Training and test data error rates for truck 103.

Figure 4.8 shows the percentage error in training and test sets for T 103. The training set consisted of (1:19650) points and the test set comprised of randomly chosen 200 points from the data not contained in the training portion. This data selection is consistent throughout this chapter, for all the other trucks and cases.

Finally, to summarize T103's result, the error rates are high at early stops – 35% in training and 40% in test. However, they converge rapidly as the number of variables in the model increases. By 18 stops, perfect classification is achieved. Of course, it is up to the user to decide the trade-off between the number of variables in the model and the resultant error rates. For consistency, we take the 21st stop as our error guide. Here it shows near perfect classification.

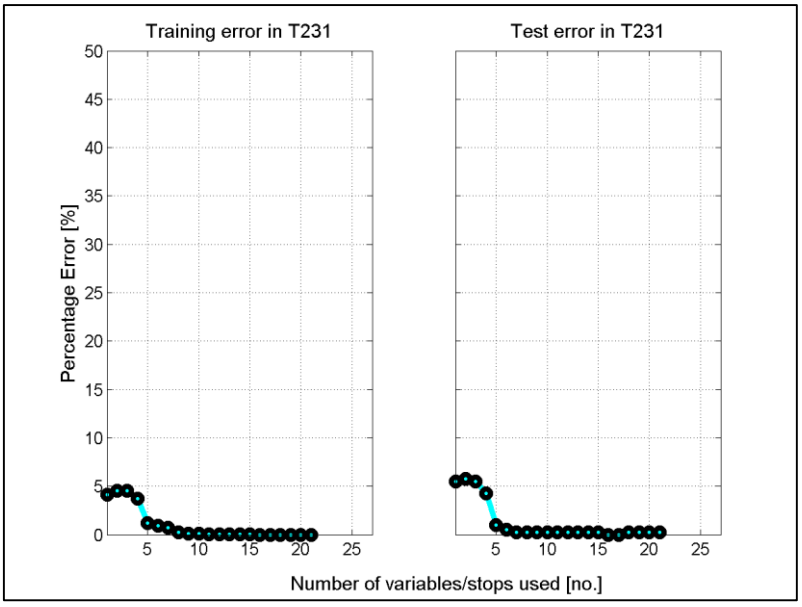

Figure 4.9: Training and test data error rates for truck 231.

T231 gives the best results amongst all the three trucks (Figure 4.9). Starting off at a low error of about 5%, it converges to almost zero by the 5th stop. This performance is strikingly good for the validation set. Figure 4.10 shows similar results for T 8572.

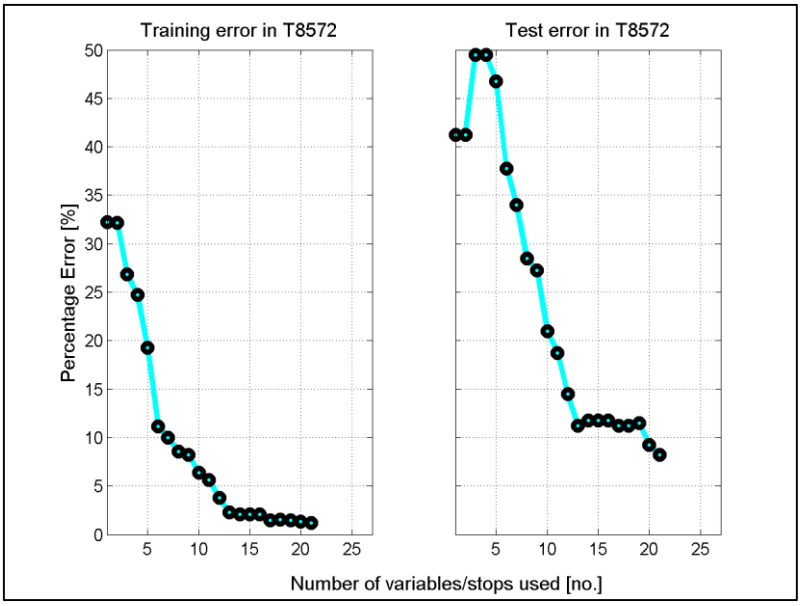

Figure 4.10: Training and test data error rates for truck 8572.

As compared to T103 and T231, T 8572 does not achieve the same levels of accuracy of classification. However, the error rates are acceptable at about zero for training and around 8% for test. Finally, Table 4.2 summarizes these results.

Table 4.2: Summary of individual truck analysis results – Set I.

Truck code	Training accuracy after 21 stops [%]	Test accuracy after 21 stops [%]
103	99.99	99.99
231	100	100
8572	98.5	92

4.8 Improvement of Quality of Data Matching

The rank-based matching method developed in [10] is a useful tool to remove selection bias from data analysis (Section 4.3). However certain limitations and a scope for improvement was identified in this work. Better matching quality was achieved in this work using the following two methods:

- Removing repeats – Rank-based data matching, as it is, allows a single healthy point to be selected repeatedly for multiple faulty points in the data. This created an imbalance in the number of distinct healthy and faulty points. The overall data-lengths were similar, but with lesser amount of distinct healthy data, this class was under-represented when visualized using a standard operation plot like a characteristic speed-torque map. Hence, removing the repeated healthy and corresponding matched faulty points was thought to be a simple and effective way of restoring the balance of distinct data within the respective categories.

- Matching quality thresholds (MQTs) – A perfect match of healthy and faulty conditions is never guaranteed. There always exist cases where even the best match of the two classes does not give a zero or close to zero difference in the values. Such a ‘bad’ best match may negatively affect the basic purpose of matching and subsequent classification. Figure 4.4 shows how the healthy-faulty differences for compressor inlet temperature are compensated using matching. However, it may be useful to remove the points that are far away from zero. MQTs allow a selection of values much closer to the absolute zero of the difference histogram.

  Two different, simple methods were used to achieve matching quality.

i. MQT method I - Predefined engine parametric filter: Drawing an analogy from the engine filters used for data selection (Section 4.2) a similar data selection filter was put together to be used post matching. These indicate selection of those healthy-faulty pairs whose differences in values for the above listed five variables, which are key signals used in matching, satisfy all the above conditions. The thresholds (listed below) were decided based on the general performance of matching on all the trucks and our intuitive sense of the difference beyond which matching could not be termed as good.

   All points satisfying the following conditions were selected for further analysis:

   - $\Delta(Engine\ Speed) \leq 50\ [RPM]$
   - $\Delta(Fueling) \leq 10\ [mg/strk]$
   - $\Delta(Fresh\ Air\ flow) \leq 1\ [flow\ units]$
   - $\Delta(Ambient\ air\ pressure) \leq 0.5\ [kPa]$
   - $\Delta(Compressor\ inlet\ temperature) \leq 5\ [deg\_c]$

ii. MQT method II - based on standard scores: This was based on the widely used, statistically driven standard score $\frac{(x-\mu)}{\sigma}$, where x is a single data point, and μ and σ are the mean and standard deviation of all the data. A standard score indicates the number of standard deviations a datum is above or below the mean. In this work, the regular definition of a standard score was slightly modified.

Since this was to be used post matching, x, a single data point is replaced with a single healthy-faulty difference value, Δ. Since the best matching quality is sought, our reference is "zero" and not μ. Finally, since five different parameters and the subsequent healthy-faulty differences are involved, it was decided to use a root-squared standard score criterion as follows:

$$\sqrt{\left(\frac{\Delta_1{}^2}{\sigma_1{}^2}+\frac{\Delta_2{}^2}{\sigma_2{}^2}+\frac{\Delta_3{}^2}{\sigma_3{}^2}+\frac{\Delta_4{}^2}{\sigma_4{}^2}+\frac{\Delta_5{}^2}{\sigma_5{}^2}\right)}<1$$

All the points or the indices satisfying this criterion were selected under MQT method 2. Individual threshold of a score of 0.5 was selected for every parameter. The root of sum of squares then becomes root of 5×0.5^2 summed five times$=\sqrt{1.25}$. Since this is an inclusive criterion unlike the previous method, poorer matching in one of the signals could be compensated by excellent matching in the others. Hence a more stringent threshold of unity is used.

Figure 4.11 compares the rank-based matching's difference histograms with the modified difference histograms after applying the MQTs individually. It is seen that the histograms become smaller in size, which is expected since the MQTs essentially are filters. It is also seen that the spread of the histogram is much smaller and points are closer to zero.

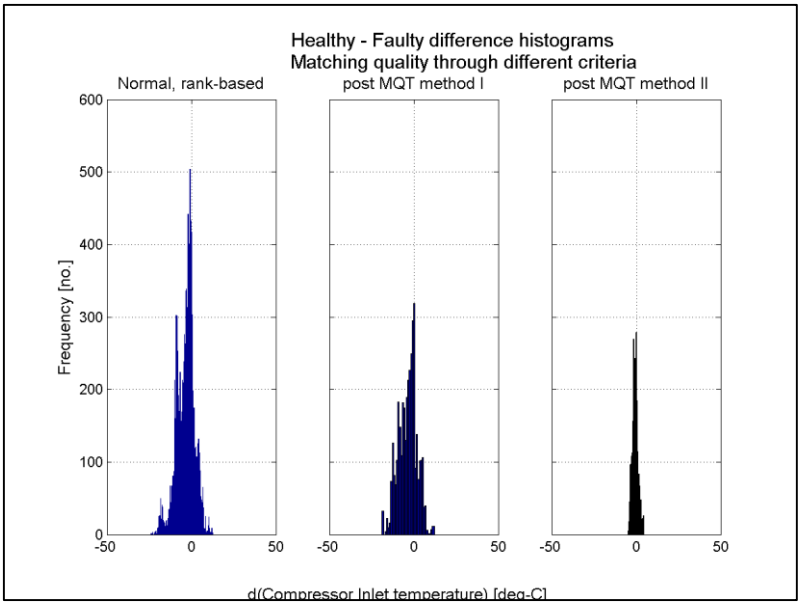

Figure 4.11: Comparing matching quality post MQTs with the actual rank-based matching using difference historagms

Through preliminary investigation, it was seen that MQT method II yielded the best classification for all the trucks when compared with MQT method I, removing repeats, and the usual matching setup used in [10]. A more effective method suggested, which is a natural intuitive extension of the results, was to apply MQT method II followed by removing repeats to ensure both quality and balance of healthy and unhealthy data. The results again were better than the standard setup. The following figures provide a visual insight for these cases applied on individual trucks.

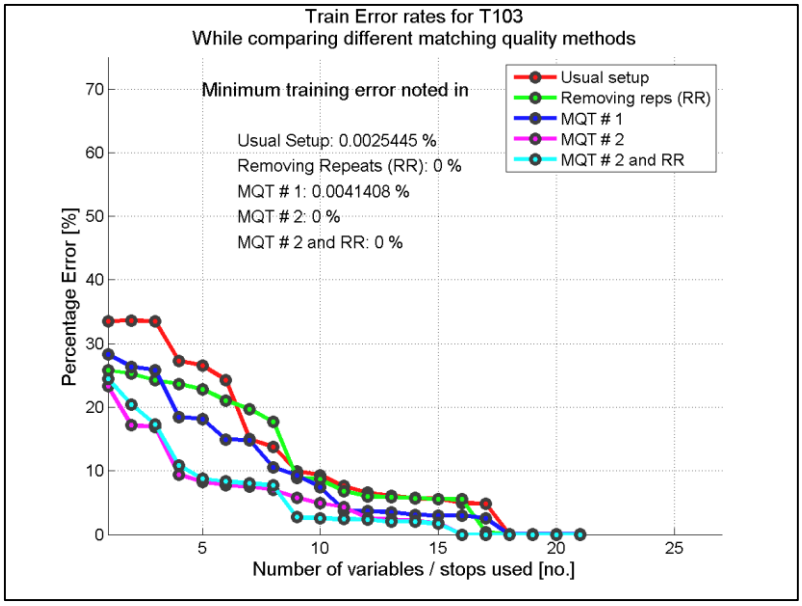

Figure 4.12: Error rates while comparing different matching improvement techniques on the training data of truck 103.

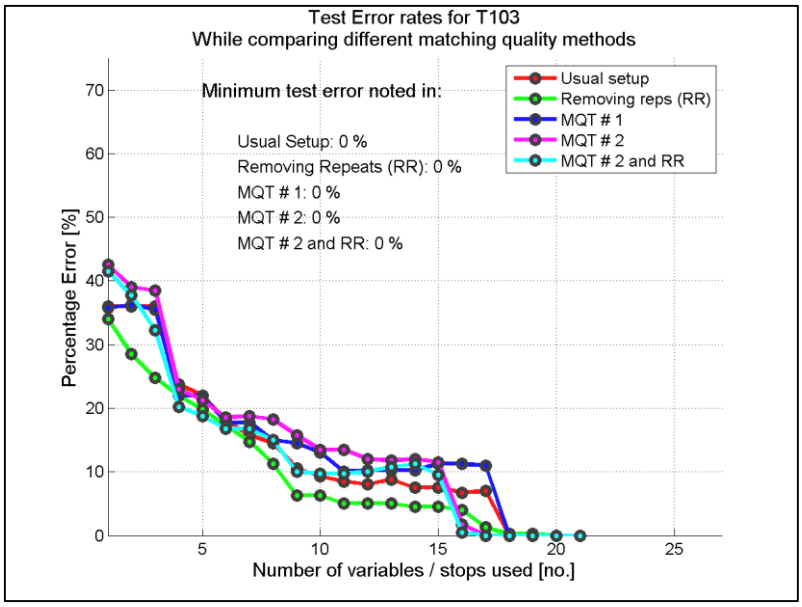

Figure 4.13: Error rates while comparing different matching improvement techniques on the common test data of truck 103.

Figure 4.12 and Figure 4.13 plot the training and test error rates, respectively, for all the previously discussed criteria for matching quality for Truck 103. As every condition is

applied, a significant number of points are removed. Hence the training sets are not of the same lengths. However, the training models of all the different criteria have been evaluated against the same test set – 200 randomly selected points as in the normal setup of T 103 data. The following figures for Trucks 231 (Figure 4.14 and Figure 4.15) and 8572 (Figure 4.16 and Figure 4.17), follow the same trend of multiple training models validated on the previously described test sets for the respective trucks.

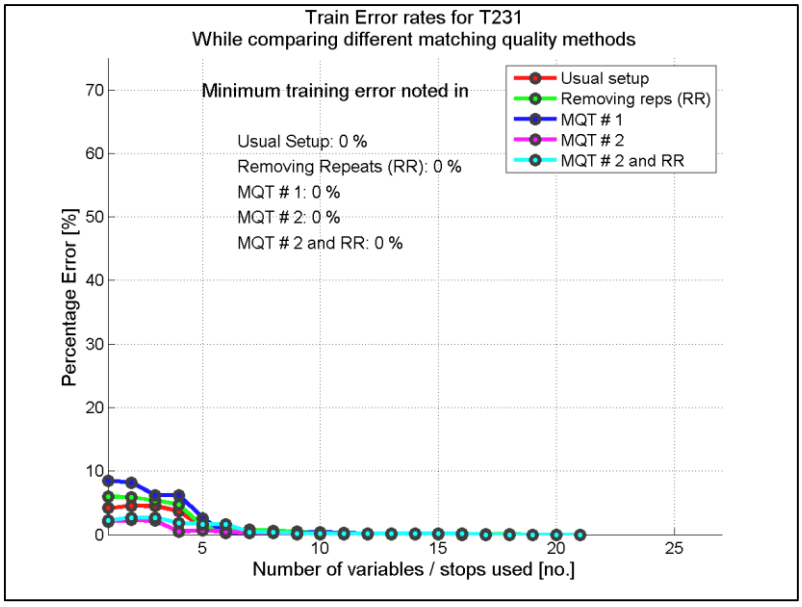

Figure 4.14: Error rates while comparing different matching improvement techniques on the training data of truck 231.

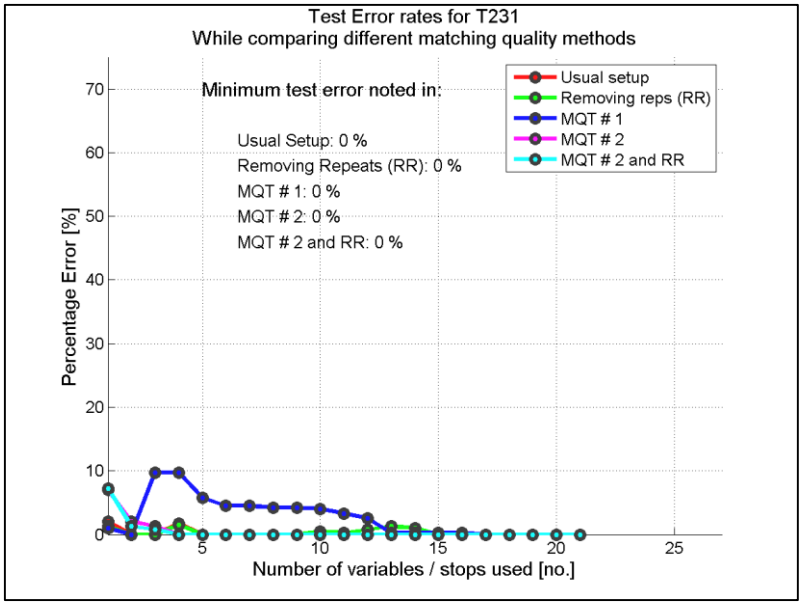

Figure 4.15: Error rates while comparing different matching improvement techniques on the common test data of truck 231.

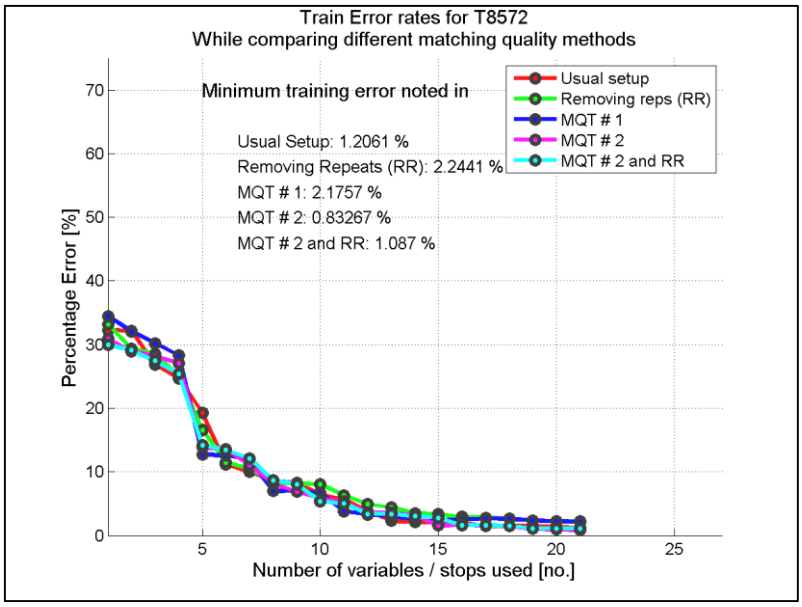

Figure 4.16: Error rates while comparing different matching improvement techniques on the training data of Truck 8572.

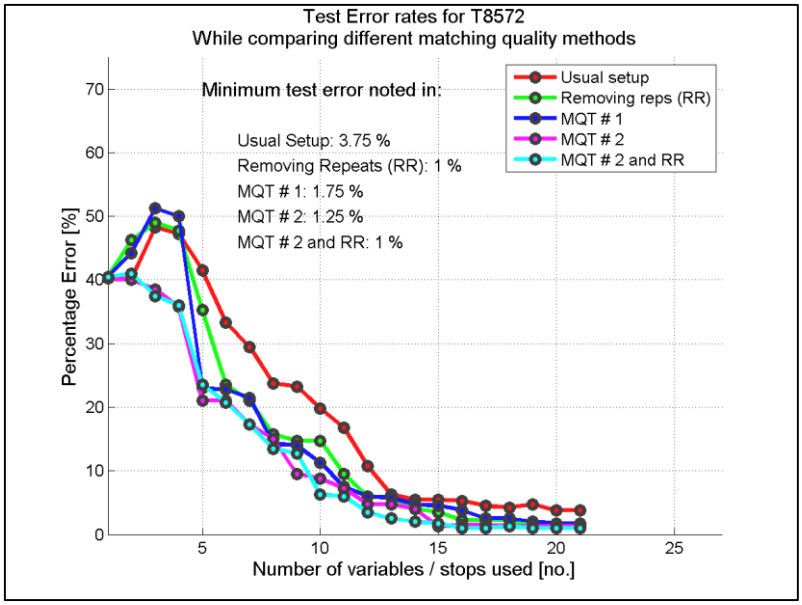

Figure 4.17: Error rates while comparing different matching improvement techniques on the common test data of Truck 8572.

Thus, the results show a marked improvement when matching quality thresholds are applied. Best results are obtained when both quality of matching is ensured and quantity of repeats is checked. As stated previously, the total number of points, with the application of these methods, decrease as well. An attrition in data may sometimes be undesirable as this may mean a decrease in the extent of training of the classifier. With reduced data, the training set may not be the best representation of various operating states that the truck goes through during the acquisition of data. This could result in insufficient learning of the classifier and would negatively affect the robustness of the classifier. This concept of training data representation will be cited in some of the other chapters and dealt with in greater detail in Appendix A.

4.9 Steady State Analysis

Automotive field test data is inherently transient in nature. This is owing to the rapid changes in modes of operation due to constantly changing road conditions. Chandrachud [8], while addressing binary classification of a 6.7 *l* engine's data obtained in a lab setup, found that a greater number of misclassifications was seen in the transient regions of data while shifting from one speed-load operating point to another. Chandrachud [8] also suggests use of multiple classifiers for multiple steady states as a possible solution to improve the accuracy of the discriminant analysis. Figure 4.18 helps visualize the effect of transience by plotting, simultaneously, the healthy (red) and faulty (blue) training data considered in this work, for T231.

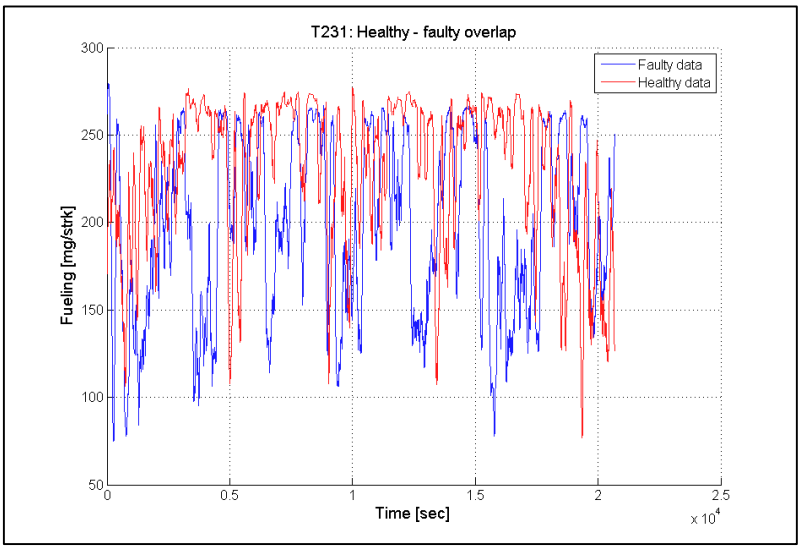

Figure 4.18: T 231 healthy faulty overlap – transience representation.

Notice the overlap of the two data streams, especially a higher degree of overlap in the regions of greater transience in just one variable. As more variables get added, the natural

healthy-faulty separation created by the presence of a fault may be affected because of the transience in the training data. This can also be visualized in two dimensions using a cross plot as shown in Figure 4.19. A cross plot is a useful tool that considers the first two variables picked by SLDA to create the best separation between healthy and faulty. The cross plot then plots the healthy training set of one against that of the other variable and similarly for the faulty counterparts. This results in two distinct clusters of data in two dimensions where healthy data is represented by blue color and faulty by red. Ideally, for 100% accurate classification, the two clusters should be completely disjoint with no overlap. In the individual classification case of T 231, EGR Position and VGT Actuator position were chosen as the first two variables in the discriminant function.

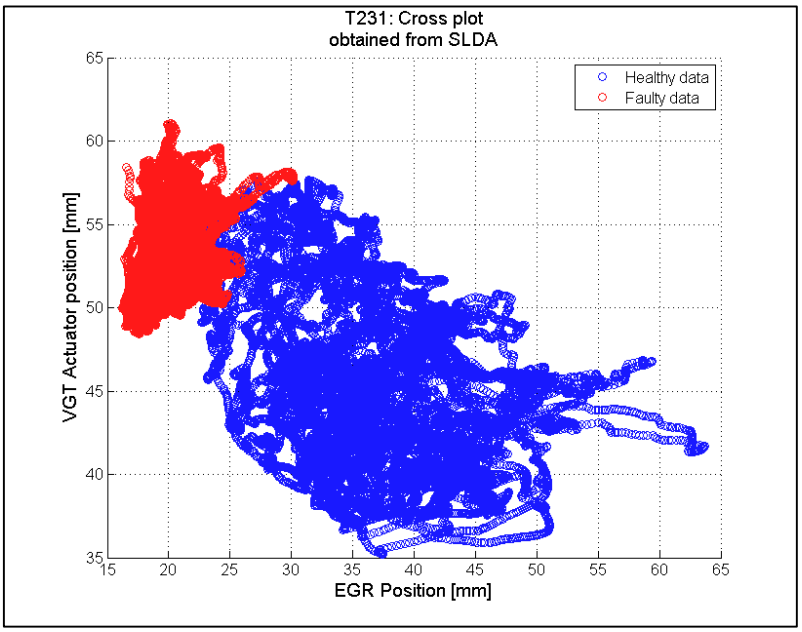

Figure 4.19: T231 cross plot – overlap between healthy and faulty sets of EGR and VGT position obtained through SLDA.

As seen from Figure 4.19, a clear overlap is seen for the healthy and faulty clusters in the cross plot. One of the possible reasons for this inaccurate classification was thought to be the transience in the training data. To counter the transience and its negative effects on the classification, we explored the potential of steady data extraction and its analysis towards the problem of discriminating faulty from healthy.

4.9.1 Steady Data Extractor

Systems and Performance engineering group at Cummins uses advanced MATLAB tools for data analysis. One such tool is 'steady finder' developed by [45]. This script scours through a day's data and extracts indices which correspond to a user-defined steady nature. The definition of steadiness requires defining three important factors:

- Definitive signals – signals based on which steady states are extracted.
- Tolerance levels – Acceptable range of every signal which bounds the steady data.
- Window size – Number of consecutive points (or seconds since sampling frequency is 1 Hz) for which the above two factors must hold.

For example, if Engine Speed with a tolerance of [+/-] 25 rpm for 30 seconds defines steadiness, the code will find indices or time instances in data where this condition was met for at least 30 seconds.

Once the indices are obtained, the steady data can be easily extracted using $Steadydata = Totaldata(steadyidx, :)$, assuming the data is arranged with variables in columns. Figure 4.20 and Figure 4.21 contain plots analogous to Figure 4.18 andFigure 4.19,

respectively, after extracting steady state data from the respective healthy and faulty sets. Clearly, the overlap between the two classes is reduced.

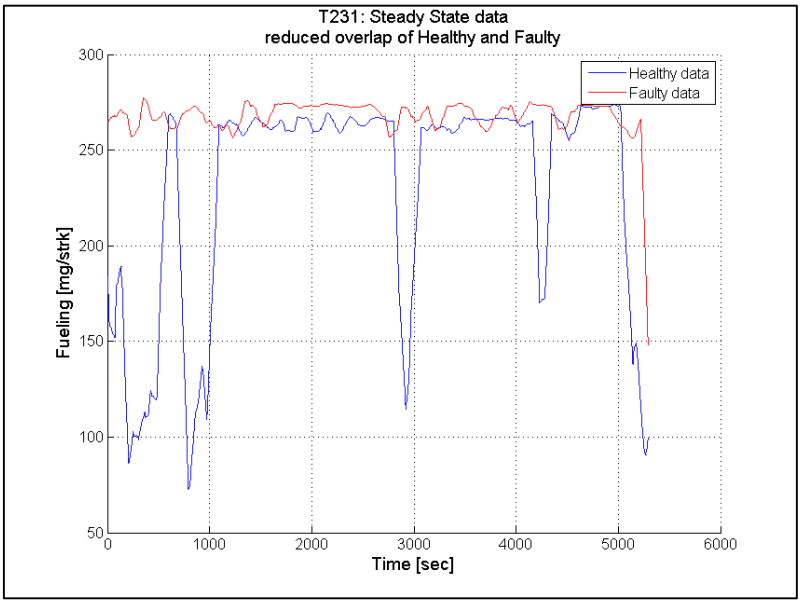

Figure 4.20: T231 – reduced overlap of healthy and faulty data.

In the cross plot obtained using only steady data (Figure 4.21), it is seen that the overlap between the healthy (blue) and faulty (red) clusters greatly reduces. This also supports the notion of eliminating the transience in data to achieve better classification.

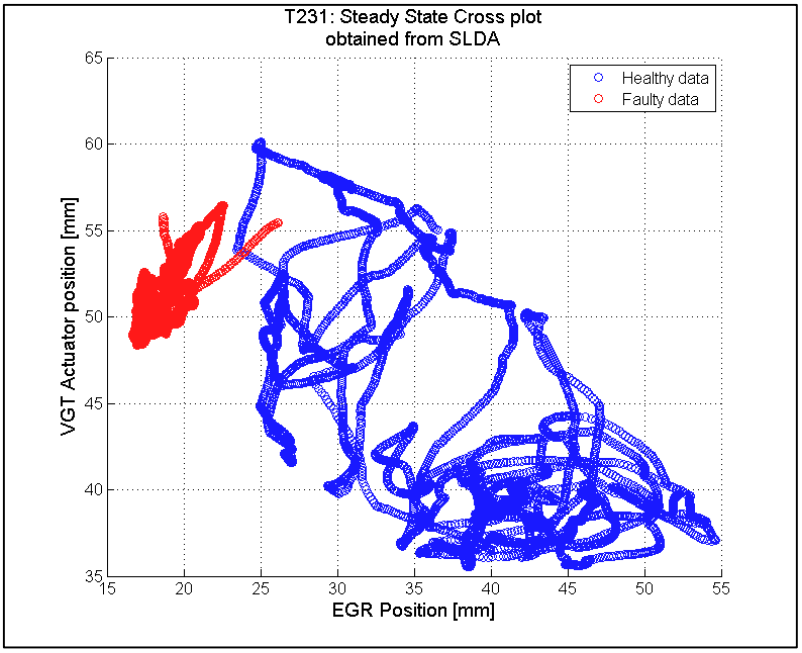

Figure 4.21: T231 cross plot – overlap between healthy and faulty sets of EGR and VGT position obtained through SLDA, but post steady data extraction.

Thus, using steady state data for classifying healthy states from unhealthy states is a potential solution to the discriminant analysis. However, this does not come without its share of performance issues. Table 4.3 summarizes the accuracies of using total data and the usual setup, and steady data using the steady state extractor.

Table 4.3: Steady vs total data accuracy of classification [%] over individual truck data.

TRAIN ↓		TEST →					
		T 103 RESULTS		T 231 RESULTS		T 8572 RESULTS	
		Tot	St	Tot	St	Tot	St
Tot		100	99.998	100	100	95.25	100
St		99.75	100	100	100	**67**	100

Both data – steady and total – perform sufficiently well to be considered further for analysis. However, when steady training is used, the accuracy of T8572 takes a dip. Such a sizable reduction in the accuracy of classification is possible if a constrained set of data, like the steady data from the steady state extractor, is used for training the classifier, and is validated against a more general case of minimally processed test data. As with the Matching Quality Threshold methods, this too, could be attributed to extent of training data representation when steady states are used to train a classifier. This is dealt with in greater detail in Appendix A. This work features a more general application to heavy-duty diesel engine data and hence does not use steady state classification in the upcoming chapters. However, based on the nature of data, steady state extraction and subsequent usage in the classification can be effectively worked out to separate good data from bad data.

CHAPTER 5. SLDA IMPLEMENTATION: MULTI-FAULT BINARY CLASSIFICATION

In the chapters leading up to this, different techniques useful for data pre-processing were described. The motivation was to remove the less useful portions of data, yet present a sufficiently large set to characterize the healthy and faulty classes, the primary application being the binary classification in individual trucks. Previously, the concatenated set for training contained only one fault, or data obtained in the presence of a single fault. It is, however, interesting to see the applicability of these pre-processing methods when more than one fault is introduced during the training phase and check the classifier performance. The classification performed is still binary in nature with two classes – healthy and faulty, i.e., no distinction is made internally within the two or more faults and all faults are tagged under one large faulty set.

Just to clarify, multiple faults essentially mean multiple trucks with a fault each, and not more than one fault belonging to the same truck at the same time. The underlying assumption of presence of only one fault in one data file still holds.

5.1 Two-Fault, Binary Classification

The first section of this chapter is dedicated to putting together a two-fault binary classifier using the data from trucks 103 and 231, and analyzing its performance using the

usual plots. Just as in the cases for individual trucks, signal and data selection (Section 4.2) and data matching (Section 4.3) are applied to every truck and data is pre-processed. The only difference in this case is the setup of $concat_{train}$ and $dummy_{train}$ matrices. These will now contain training data and encoding corresponding to both trucks. Analogies can still be easily drawn since the only change is that the overall number of points (or the number of rows) of $concat_{train}$ and $dummy_{train}$ is twice as large as compared to the individual case. As mentioned previously, the same number of points, i.e. 19650, of every class of every truck is used here too. Hence, the $concat_{train}$ and $dummy_{train}$ matrices would look as follows:

$$concat_{train} = \begin{bmatrix} {xh_1}^1 & \cdots & {xh_1}^{21} \\ {xh_2}^1 & \dots & {xh_2}^{21} \\ \vdots & \ddots & \vdots \\ {xh_{39300}}^1 & \cdots & {xh_{39300}}^{21} \\ {xf_1}^1 & \dots & {xf_1}^{21} \\ {xf_2}^1 & \dots & {xf_2}^{21} \\ \vdots & \ddots & \vdots \\ {xf_{39300}}^1 & \dots & {xf_{39300}}^{21} \end{bmatrix}, \text{ and } dummy_{train} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1_{39300} & 0_{39300} \\ 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 0_{39300} & 1_{39300} \end{bmatrix}$$

There are, in all, four truck-class pairs, viz. T103-Healthy, T103-Faulty, T231-Healthy, and T231-Faulty. 19650 points are picked from each of the pairs as described in detail in Chapter 4. The encoding in the $dummy_{train}$, subsequently doubles in size too. Test data, again, is kept separated from the training and is built similar to the individual case – by picking 200 random points from the remainder of datasets of every truck-class pair. This

training and test setup is evaluated using the SLDA-classify.m code and results are populated as below. The other specifications of number of variables introduced in the model, priors, the lambda value and maximum number of iterations are unaltered owing to the binary nature of classification.

It is seen in Figure 5.1 that although the convergence to minimal error is slower than the individual classification of each truck the error rates go down to lesser than 5% by 21 stops. The performance of the test set is slightly less accurate as expected, when compared with its training counterpart. It is surely above the acceptable lower threshold of accuracy.

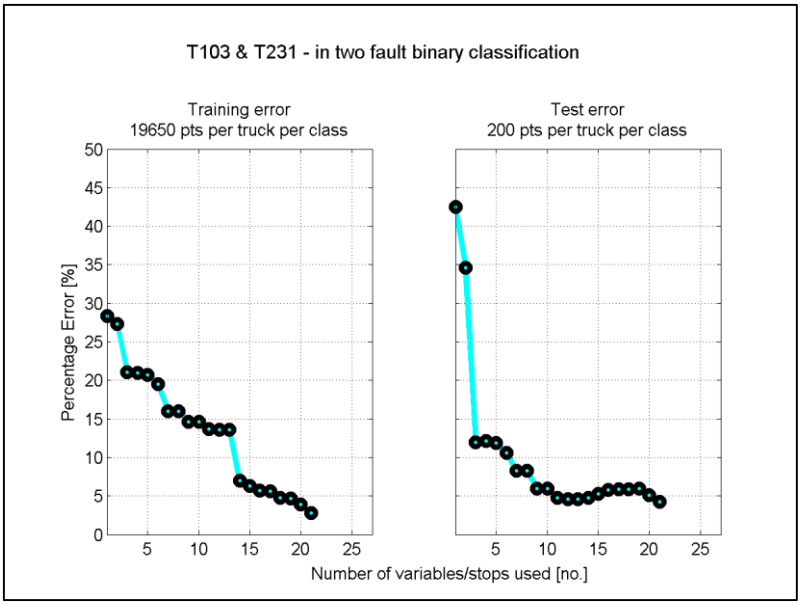

Figure 5.1: Training and test error rates for T103 & T231 two-fault, binary classifier.

Figure 5.2 highlights the instances of misclassification. Using this plot in the multi-fault case gives an intuitive understanding of the ‘ease of classification’ that each truck offers to the SLDA.

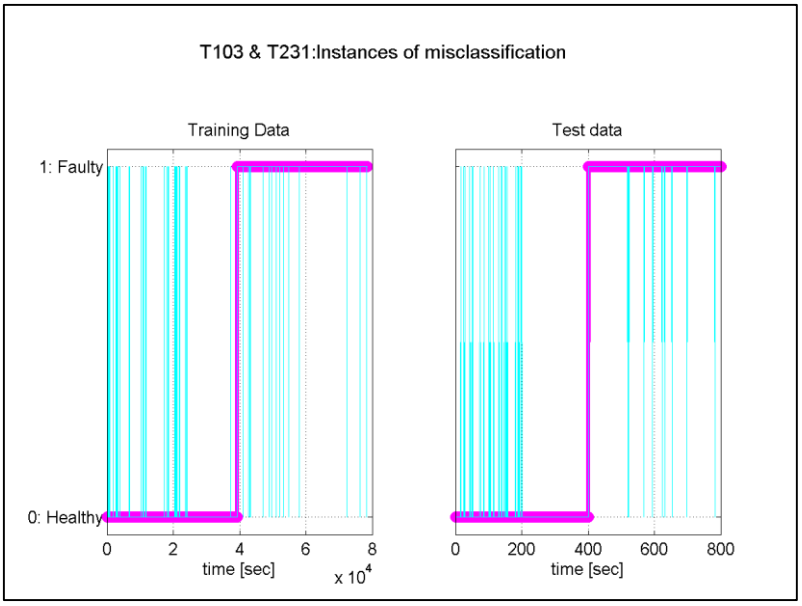

Figure 5.2: Instances of misclassification in the T103 and T231's binary classifier.

As described previously, there are four truck-class pairs in a two-fault binary classifier, and each pair consists of 19650 data points. The overall healthy data is a concatenation of T103-Healthy and T231-Healthy, in that sequence, and similarly, the overall faulty data is a concatenation of T103-Faulty and T231-Faulty. Hence, for every zone (0:Healthy, 1:Faulty) in Figure 5.2, the first 19650 points belong to Truck 103 and the latter half to Truck 231. Similar pattern is followed in other multi-fault classifiers.

Truck 103 shows a high number of misclassifications while Truck 231 does exceedingly well (second half of both healthy and faulty levels) in avoiding misclassifications of both kinds – healthy classified as faulty (false positives) and faulty classified as healthy (false negatives). This indicates a better characterization or a clear separation of the T 231 fault. T 103, on the other hand, is not clearly disjoint from the healthy set. As will be seen in the next chapter that there does exist a region of overlap between the healthy data and T103's

faulty data. This region causes the number of misclassifications in T103 to be higher than in T231.

5.2 Three-Fault, Binary Classifier

The encouraging results of the two-fault classifier motivated a natural extension to a more complex combination of three trucks (or three engines) with distinct hardware, calibrations and power ratings. Please refer to section 4.1 for more details. Just as for the previous case, three different fault modes are used here in a binary classification setup. Again, the motive is not to pinpoint a fault location or its nature but to identify anomalous behavior within the concatenated data.

The addition of a third truck to the two-fault classifier has a very small effect on the overall setting up of the data. In this case, too, 19650 data points are selected from each truck-class pair. The only difference is the total number of points now is $19650 \times 6\ (truck - class\ pairs) = 117{,}900$. Test data is also constructed similarly as in the previous case. Since the structure is still binary in nature, priors, lambda and maximum iteration values are kept the same as for the previous cases. The next two figures (Figures 5.3 and 5.4) show error rates and instances of misclassification in the three-fault case. Interpretations similar to the previous case can be drawn here too.

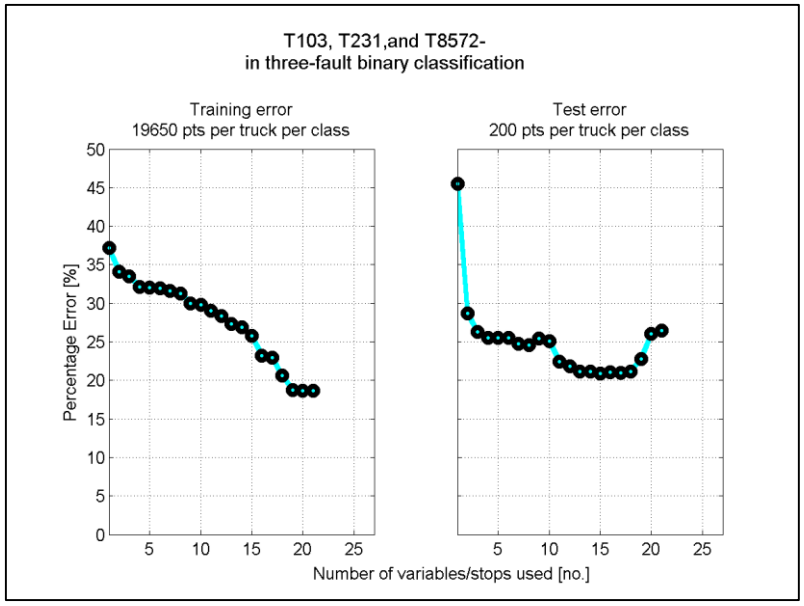

Figure 5.3: Training and test error rates for Trucks 103, 231 and 8572's three-fault classifier.

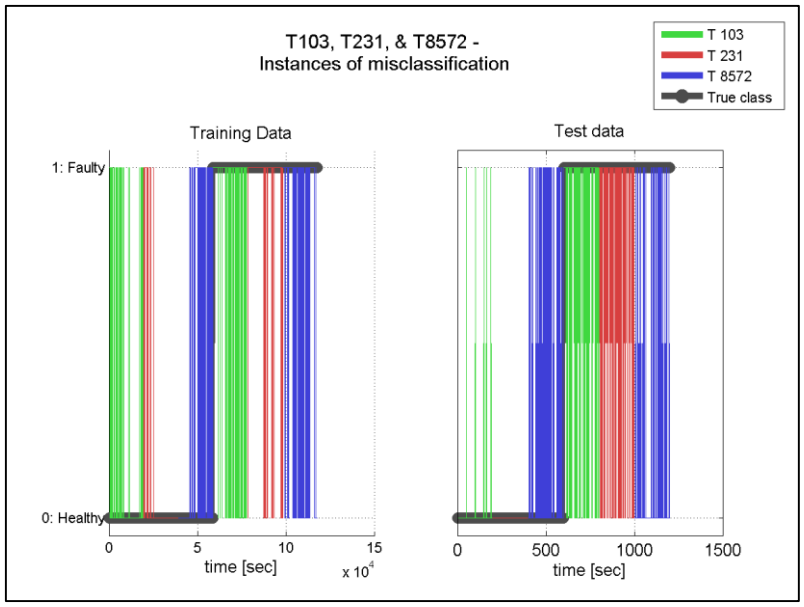

Figure 5.4: Instances of misclassification in the three-fault classifier of T103, T231 and T8572.

A point to note (Figure 5.3) is that with the introduction of T8572, the overall classification accuracy decreases manifold. The training accuracy suffers a drop to 82% while the test

accuracy is also low at 80%. This is also corroborated by the "instances of misclassification" plot (Figure 5.4).

The results suggest a potentially larger overlap in the healthy and faulty regions of T8572, indicating an unclear separation. An investigative study was conducted which led to the inspection of the raw data sets of the three trucks. It was found that histograms of T8572 bore a striking dissimilarity from the other two trucks. There were only a few regions where the three trucks experienced similar operating conditions. T103 and T231 were still relatively under similar operations. This led to the proposition of using only the trucks that have similarities in their operating conditions and duty cycles. This further motivated checking both the performance of SLDA and the validity of the claim that 'similar trucks do better' by making a new classifier using the T200 series which, according to the pedigree chart, were similar to each other and also had a malfunction of the same hardware origin.

5.3 T 200 Series Classifier

Until now, the multi-fault classifiers designed contained trucks from very distinct platforms with engines completely different from one another. It was, therefore, of interest to see if trucks belonging to the same fleet with engines with similar calibrations and hardware will make for a more accurate classifier. As described in Section 4.1, T200 series – T231, T232 and T234 – are fit for this purpose. Since these trucks had fault modes originating from similar hardware (EGR cooler malfunction), better results were expected.

The data was set up in the same manner as the previous three-fault classifier with 19650 points picked from every truck-class pair. Other parameters take the same values as before, and the two sets of plots (Figures 5.5 and 5.6) convey the results.

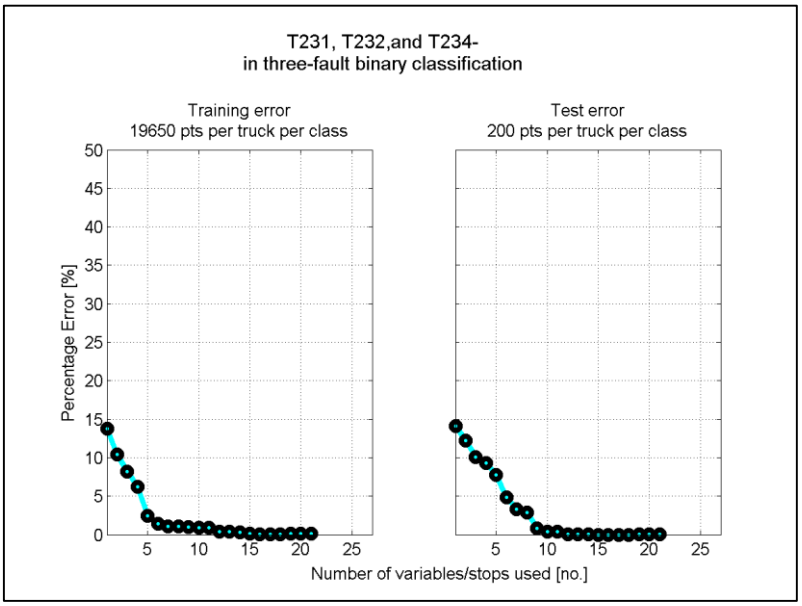

Figure 5.5: Training and test error rates for a T200 series binary classifier.

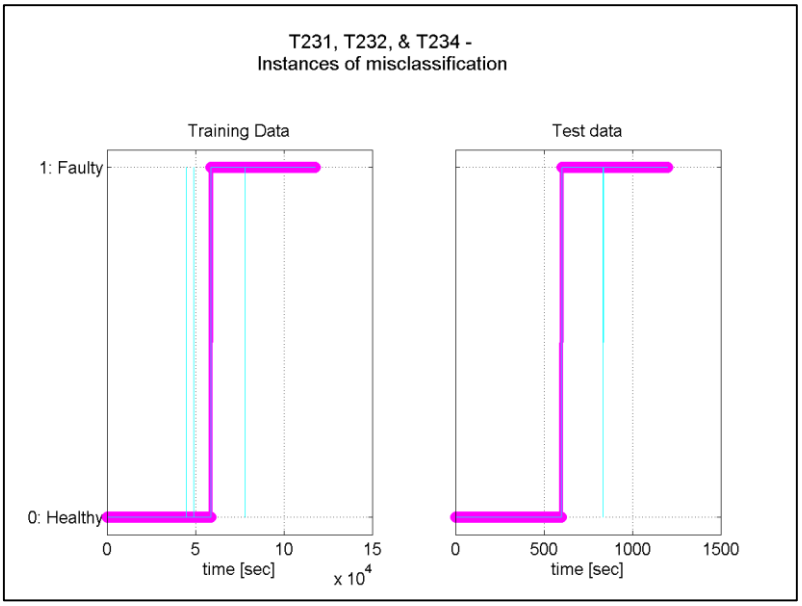

Figure 5.6: Instances of misclassification in T200 series classification data.

Considering this classifier involved large data from three trucks, the results as seen from the figures are excellent. The training and test error converge rapidly to zero, with only half the maximum number of stops. Instances of misclassification are also very low in both training and test sets. These results support the hypothesis that SLDA finds it easier to discriminate and classify the data belonging to similar engines or engines experiencing similar operating conditions. Table 5.1 summarizes the results of the classifiers described in this chapter.

Table 5.1: Summary of multi-fault, binary classifier performance.

Classifier	Training accuracy [%]	Test accuracy [%]
Two-fault (T103, T231)	97.5	96.5
Three-fault (T103, T231, T8572)	81.5	80
T200 series	99.9	99.9

CHAPTER 6. SLDA IMPLEMENTATION – MULTIPLE CLASSES OF DATA

Chapters 4 and 5 discussed SLDA from the perspective of binary classification. Even if multiple faults were treated, they all were categorized as one class. This chapter rounds out the various implementations of SLDA with a more general example, the instance when all faults are treated separately, and where every fault is represented as a different class of data. This is done for three faults – T103, T231 and T8572 – and hence for a classifier comprising of four classes (three faults and one overall healthy set). The small but important changes in the parametric values and the resultant usual plots are described. Two different test sets are used to establish the validity of this classifier. These validation cases help satisfy both research objectives enumerated in Chapter 1 – effective characterization of healthy data and efficiently allocating the faulty data to the outlying faulty cloud.

6.1 Three-Fault, Multi-Class Classifier

Most of the logic, programming and mathematics remain similar to the previous cases of SLDA implementation. Changes to some of the variables, guided by reason and intuition, ensure that the algorithm is modified to suit the multi-class problem. These changes are described below:

- $dummy_{train}$: Previously, the $dummy$ matrix had two columns of zeroes and ones arranged in a pattern that reflected the location of healthy and faulty data in the $concat_{train}$ matrix (Chapter 4.4). The two columns resulted because of the two classes. With four different classes, the $dummy_{train}$ assumes a structure of four columns. Atypical though it is compared to the usual two-class arrangement, it is quite intuitive and self-explanatory. The structure is shown in a MATLAB program format, since it is easier to represent and understand it this way as compared to matrix representation. The interjecting comments explain different steps of the program:

```
train_size = 19650; % number of points from each
truck-class pair of training data

onestrain = ones(train_size,1); % declaring a column
of 1s

zerostrain = zeros(train_size,1); % declaring a column
of 0s

%All healthy data put together in one variable
train_healthy = [truck103healthy(train_size,:);
                 truck231healthy(train_size,:);
                 truck8572healthy(train_size,:)];

%Different variables for every faulty set
truck103faulty = truck103faulty(train_size,:);
truck231faulty = truck231faulty(train_size,:);
truck8572faulty = truck8572faulty(train_size,:);
```

```
%concat_train matrix where healthy and faulty data is
concatenated.
concat_train = [train_healthy;
truck103faulty; truck231faulty; truck8572faulty];

%defining columns of the dummy_train matrix. Note the
length and position of ones coincides with a
particular class in concat_train
dummy_train(:,1) = [onestrain; onestrain; onestrain;
zerostrain; zerostrain; zerostrain];
%Since overall healthy data is thrice in length as
compared to each faulty set, using onetrain thrice to
mark encode the healthy data
dummy_train(:,2) = [zerostrain; zerostrain;
zerostrain; onestrain; zerostrain; zerostrain];
dummy_train(:,3) = [zerostrain; zerostrain;
zerostrain; zerostrain; onestrain; zerostrain];
dummy_train(:,4) = [zerostrain; zerostrain;
zerostrain; zerostrain; zerostrain; onestrain];
```

A careful observation of the code and the pattern in which onestrain and zerostrain are arranged to mark different categories of data brings out the intuitive picture of the overall concatenation and encoding in the dummy. This arrangement ensures that every class is represented adequately in the algorithm.

- *'group'* matrix: Previously the *'group'* contained two indicator variables in a column to represent two classes. Now it contains four. As said before, the indicator column need not be binary or numeric but could contain an alphanumeric header/marker for every class. In this work and chapter, *'zero'* is the marker for

healthy data, *'ones', 'twos', 'threes'*, respectively, are the indicators for *faulty103train, faulty231train and faulty8572train*. So, in the MATLAB program, the group matrix is declared as

```
group = [zerostrain; zerostrain; zerostrain;
         onestrain; twostrain; threestrain];
```

`twostrain` and `threestrain` are defined exactly as `zerostrain` and `onestrain`. The length of every individual entry and each faultytrain vector is 19650.

- prior: priors need to be expressed as a vector of length equal to the number of classes and entries that represent the *a priori* probability of each class. The faulty data is classified into three different groups as described through the dummy and group variables. This forces the priors to be biased to healthy since now the length of each faulty class is a third of the length of the overall healthy class. By the fractional rule, $p(n_1) = \frac{n_1}{n_1+n_2+n_3+n_4}$, the priors are obtained as a vector [3 1 1 1].

These changes are adopted in the algorithm for the multi-fault problem. The other aspects of signal and data selection, data matching and results are the same as in the previous chapters. In this case too, total data is used without applying additional processing filters and the test data setup is similar to that in all the previous cases. The results are shown below using the customary plots for error rates and instances of misclassification.

It is seen from Figure 6.1 that the overall error rates for training and test data for the three-fault classification are considerably lower in this case compared to the three-fault binary classification described in the last chapter. Intuitively, this makes more sense too since the

faulty data of one truck is quite different from the other and putting it in separate classes could lead to better characterization and hence a better solution.

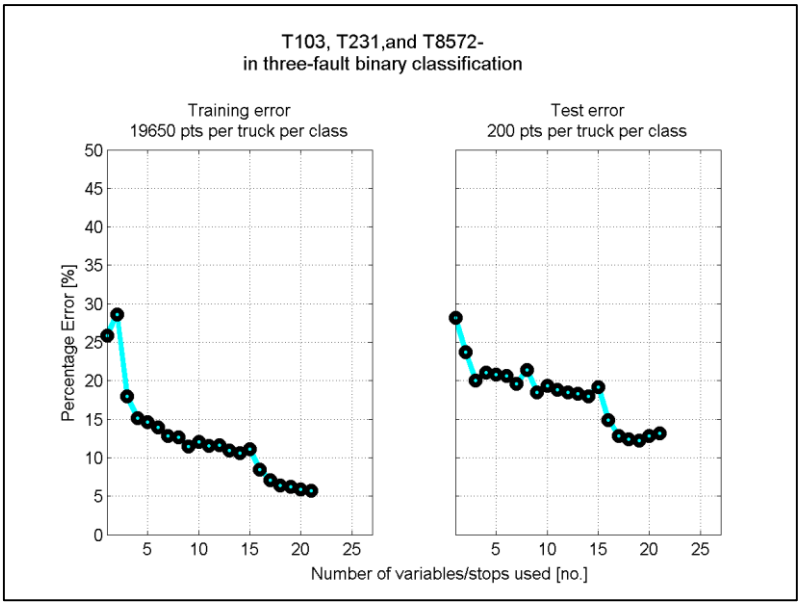

Figure 6.1: Error rates for three-fault (T103, T231, and T8572), multi-class solution to SLDA.

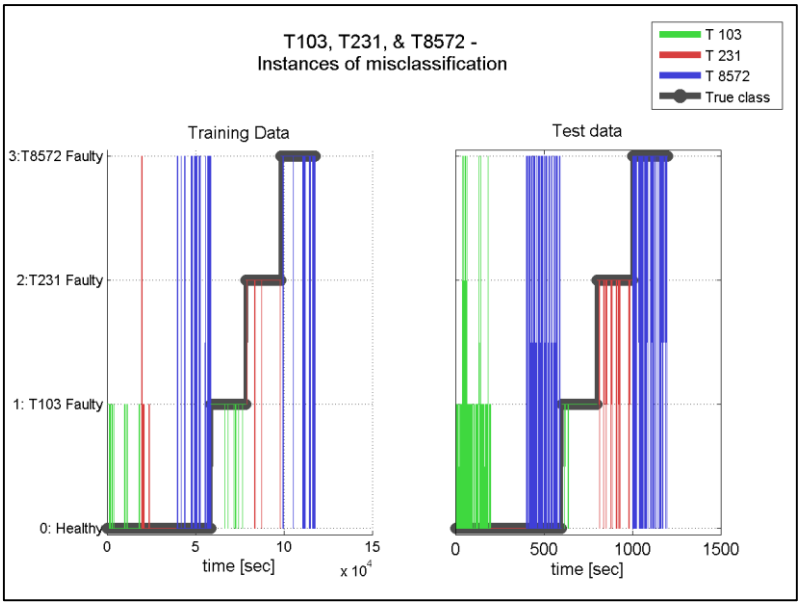

Figure 6.2: Instances of misclassification in the three-fault, multi-class classifier of T103 , T231 and T8572.

The plot, “instances of misclassification” (Figure 6.2) now has four levels 0 to 3 to represent the four classes. It is clearly seen that T103 and T8572 are more difficult to classify, whereas T231 is easily separated into respective categories. The plot also shows a marked reduction in misclassifications as compared to binary classification.

As discussed in Section 3.3, SLDA offers a maximum of “k-1” sparse directions to visualize low-dimensional views of a “k” class training data. This attribute is very useful in a multi-class problem because these images give great insights into data characterization and separation. Since the problem now has four classes, three orthogonal directions are obtained to get a 3-D perspective of the projections. The training data for this classifier is manifested using these directions as follows (Figure 6.3).

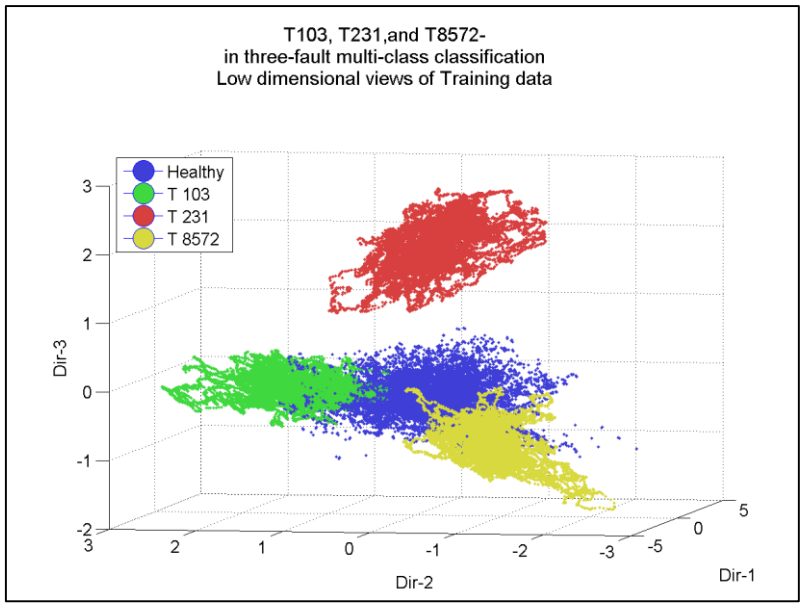

Figure 6.3: Low-dimensional projection view of training data in a three-fault, multi-class classifier.

The central blue cloud represents the healthy data and the surrounding clusters represent the three faults. As seen from the figure, T231 cluster is clearly well separated from the rest of the data and thereby proves its clear characterization. T103 and T8572 find their healthy and faulty data overlapping to a minimal extent. The three directions obtained from SLDA are orthogonal in nature and thus give a good perspective of the projected data. The plot of Figure 6.3 when viewed from the Dir-1 – Dir-3 plane, is seen in Figure 6.4. This side view will be useful when the three-fault, multi-class classifier will be validated against brand new, completely unseen test data.

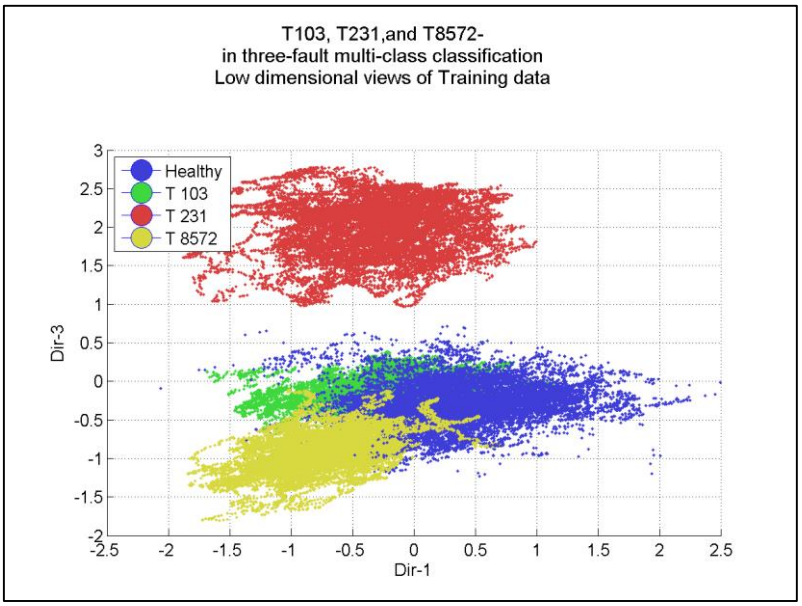

Figure 6.4: Side view of the low-dimensional projections of training data.

6.2 Validation Using Five New, Healthy Trucks

From the individual classification in Chapter 4 to the three-fault multi-class classification described in this chapter, the test set was always a part of the overall data of the trucks. In February 2013, Cummins provided five new data sets belonging to ISX 15*l* family, all

without anomalies, belonging to the same platform, hardware configuration, calibrations and similar duty cycles and power ratings. Different ambient conditions were represented through the five sets. The summary of data for these trucks is shown in Table 6.1.

Table 6.1: Healthy truck power-torque pedigree chart.

Truck / File name	**Power (hp)**	**Torque(lb-ft)**	**Ambient temperatures and pressures**
TruckA.mat	450	1650	Sea Level, Hot Temps
TruckB.mat	425	1750	~200 [m], Hot Temps
TruckC.mat	425	1750	0-800 [m], Moderate Temps
TruckD.mat	450	1650	500-2000 [m], Cool Temps
TruckE.mat	450	1750	200-400 [m], Cold Temps

The primary motive of this exercise was to get a reality check to see if the algorithm could categorize all or most of the above data as healthy using the aforementioned three-fault classifier. The validation would now be successful if the five healthy known sets are engulfed by the blue cloud. Achieving this would satisfy the research objective in the first place and establish a working design of the technique.

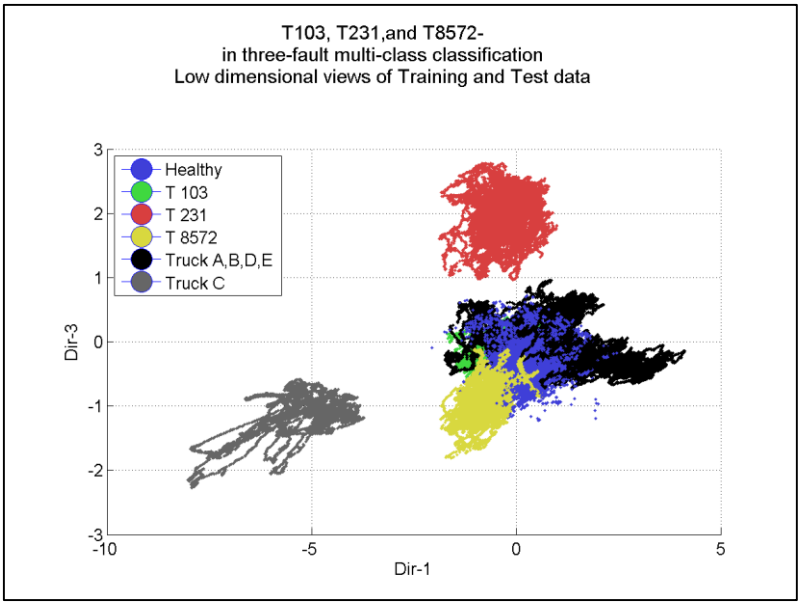

Figure 6.5: Representation of new, healthy truck data in 3-D space defined by the sparse directions obtained from SLDA of three-fault, multi-class classifier.

For validation, data of the five trucks were concatenated together as a large test. The 3-D view of the entire data (Figure 6.5) is quite intriguing. All the trucks were expected to be engulfed in the blue central cloud. Four out of five follow the intuition, while one – Truck C – does not. Truck C's entire data is completely disjoint from the busy area of training clouds. To investigate this, plots of raw, unprocessed data were inspected. All the 21 variables selected in the model were plotted. For immediate comparison, the five data sets were concatenated so that differences in data could be easily captured visually. Figure 6.6 shows a collection of such plots. Here trucks A, B, D, and E are in blue and truck C's data is in red.

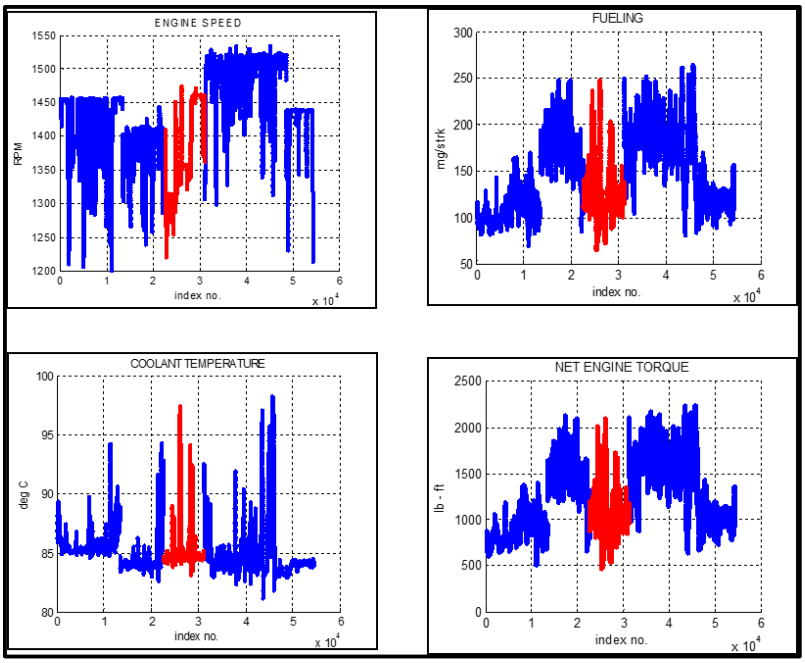

Figure 6.6: Engine speed, load, fueling, and coolant temperature for concatenated data of five healthy validation trucks.

Even though five data sets are distinctly visible, no clear trends are seen in Figure 6.6 to suggest an abnormality in truck C.

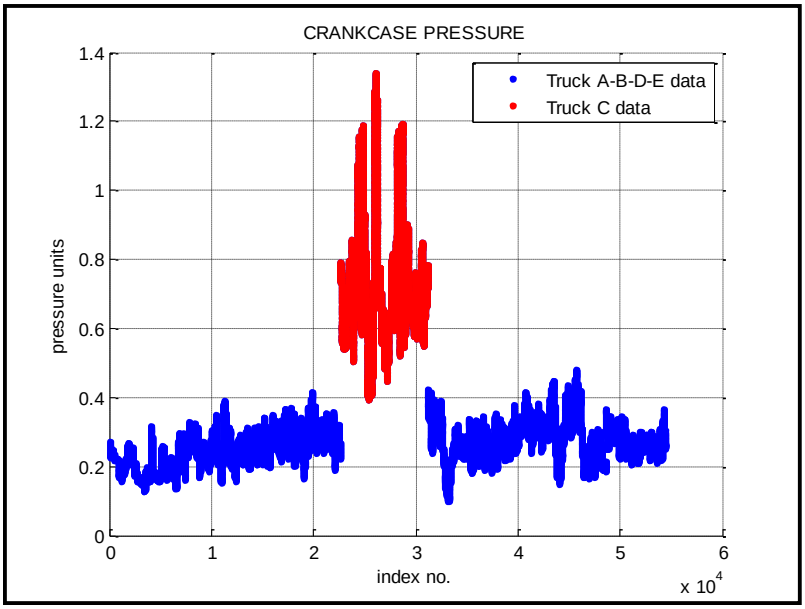

Figure 6.7: High mean and deviations manifested by crankcase pressure of truck C.

This held for all the variables except crankcase pressure as shown in Figure 6.7. The mean and standard deviation of the crankcase pressure in Truck C's case are much higher than the rest of the data. Since the results do not explain the nature or location of the anomaly, more investigation is needed to establish the causal relations of high crankcase pressure in truck C. It will be of much interest to look carefully at some of the data of truck C that was collected a few weeks later in time with respect to the current data to see if there was an increase in the mean and standard deviation of the crankcase pressure of this truck, potentially nearing a faulty state.

Consultations with Cummins engineers yielded that even if Truck C's crankcase pressure was not bordering on the abnormal operating region, it was certainly different from the other normal operation of similar trucks. It was agreed that this may not have triggered a fault at a later stage, but equally agreed upon was the observation that SLDA could capture deviations from relative means and had the potential to be developed into a tool of greater predictive powers.

The simple fact that SLDA could flag a truck whose data was deviating from the relative mean was a significant achievement. This clearly proves that even with non-extensive training, the algorithm picks up minimal differences in data. This capability of SLDA can be potentially utilized to predict future occurrences of faults or abnormal behavioral changes in the data. These results are greatly motivating to pursue this work further towards developing a health monitoring system for field-test diesel engines.

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

This work was intended to utilize the enormous data at hand and devise a simple, robust and accurate method to cluster healthy data and separate it from the clouds of anomalous data towards achieving the prime objective of identifying anomalous behavior in field-test trucks. Founded on some of the earlier work in the group, this work aimed to push the bar higher from a single fault detection mechanism to multi-fault analysis. That Sparse Linear Discriminant Analysis can be adapted as a fault detection and diagnostic method has been successfully manifested in this work. Important findings and conclusions of this work are enumerated below.

1. Data selection and matching techniques were utilized and improvised for a comprehensive training process and removal of selection bias, if any.
    a. Steady state data and matching quality control methods can be used to improve the performance of classification. Their selection and application should be driven by the nature of the problem and the nature of the data at hand.

   b. While using these additional data processing methods, test errors tend to converge not as rapidly as their training counterparts. Subsequent study in the overall "representativeness" of the training data (Appendix A) help explain this pattern. According to the findings, focused training data with narrower coverage of operating regimes will struggle to classify a more general, less processed test data set.

2. Similarities in engine architecture, engine ratings, calibrations, etc. aid the overall performance of the algorithm while solving a multi-fault (essentially multi-truck) classification problem.
3. In the case of distinct, multiple trucks with dissimilar engine configurations, it is more advantageous to define a multi-class training data.
   a. Distinct faults are better characterized leading to improved accuracy compared to that of a binary classifier.
   b. Such a model offers insightful 3-D perspective of the data detailing clustering and regions of overlap (where misclassifications tend to be higher).
4. With even a basic training model, general healthy performance characteristics are well captured and even small distinctions are flagged for further check.

7.2 Contributions

This work was developed from some of the fundamental research carried out while solving similar problems at Purdue University. However some of the exclusive contributions of this work are enumerated as follows:

1. Validation of SLDA towards addressing the problem of detecting anomalous field data using a multi-fault, multi-class scheme for classification and brand new, hitherto unseen data of five healthy trucks.
2. Generation of orthogonal sparse directions and the resultant three-dimensional space to visualize the complex data with twenty-six parameters.
3. Introduction to steady state training models and methods to improve matching quality in the data processing phase of the algorithm. This was performed with a view to improve the overall accuracy of classification.
    a. Provided visual insights into the "representativeness" of the training data and its relation with the convergence of test errors.

7.3 Future Work

This work has its current limitations of scope and hence great promise for some worthwhile future research.

Immediate future research could involve working on the trade-offs introduced in this manuscript. Specifically, focus could be on the choices concerning the selection of a data pre-processing technique and the associated trade-off with the coverage of operating regimes provided by the reduced data.

From an industrial perspective, one of the main concerns is the scale of application. A classifier which has been devised using data from three to five trucks cannot be practically utilized to analyze tens and hundreds of vehicles. To this end, multiple training models can be generated to analyze different classes of data based on the fleet and engine ratings, calibrations, ambient conditions, driver choices and even the season of data collection. This

will help ensure removal of bias and subsequently lead to better data fits, and fewer instances of over fitting in the models.

With the developments in modern OBD systems, it is imperative to search for ways to incorporate this model in the ECM of the engine for immediate training, live analysis and a quicker corrective action.

Finally, this process-history based method, at the moment, does not address the causality of faults which poses a serious challenge to the service department of any manufacturing unit. Blending in the physics of the system to determine the cause of anomalous behavior can be a challenging but fruitful exercise.

LIST OF REFERENCES

LIST OF REFERENCES

[1] C. Curtis, "President Obama Announces First Ever Fuel Economy Standards for Commercial Vehicles," 2011.

[2] CARB, "On-Board Diagnostics (OBD) Program," 2013. Available: http://www.arb.ca.gov/msprog/obdprog/obdprog.htm.

[3] Ravindra, V. Kakade, "Fault Detection Using Spectral Methods: Wavelets and Correlation Techniques", Masters Thesis, 2011.

[4] A. Joshi, P. Deignan, P. Meckl, G. King, and K. Jennings, "Information Theoretic Fault Detection", Proceeding of the American Controls Conference, pp. 1642–1647, June 8 – 10, 2005.

[5] P. B. Deignan, G. B. King, P. H. Meckl, W. Lafayette, and K. Jennings, "Confidence Measure Estimation in Dynamical Systems Model Input Set Selection", Proceeding of the American Control Conference, pp. 2824–2829, June 30 – July 2, 2004.

[6] Alok A. Joshi, "Strategies for Data-based Diesel Engine Fault Diagnostics", Doctoral Dissertation, Purdue University, 2007.

[7] Alok. A. Joshi, S. James, P. Meckl, G. King, and K. Jennings, "Assessment of Charge-Air Cooler Health in Diesel Engines Using Nonlinear Time Series Analysis of Intake Manifold Temperature.", Journal of Dynamic Systems, Measurement, and Control, vol. 131, no. 4, May, 2009.

[8] N. Chandrachud, "Classification of the health of diesel engines using Sparse Linear Discriminant Analysis",Masters Thesis, Purdue University, 2009.

[9] L. Clemmensen, T. Hastie, and B. Ersboll, "Sparse Discriminant Analysis," 2008.

[10] B. A. Warman, "Data Analysis of Diesel Engine Faults", Masters Thesis, Purdue University, 2012.

[11] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, Springer-Verlag, Berlin, 2006.

[12] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri "A Review of Process Fault Detection and Diagnosis Part I: Quantitative Model-Based Methods," *Computers and Chemical Engineering*, vol. 27, no. 3, p. 293–311, 2003.

[13] M. Nyberg and L. Nielsen, "Model Based Diagnosis for the Air Intake System of the SI-Engine." Research article, Vehicular Systems ISY, Linkoping University, Sweden, 1997.

[14] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A Review of Process Fault Detection and Diagnosis Part II: Qualitative Models and Search Strategies", *Computers & Chemical Engineering*, vol. 27, no. 3, p. 313–326, 2003.

[15] V. Venkatasubramanian, R. Rengaswamy, K. Yin,and S. N. Kavuri, "A Review of Process Fault Detection and Diagnosis Part III: Process History Based Methods", *Computers & Chemical Engineering*, vol. 27, no. 3, p. 327–346, 2003.

[16] I. A. Kakadiaris, "Physics-Based Modeling , Analysis and Animation Physics-Based", *Technical Reports (CIS)*, University of Pennsylvania, 1993.

[17] J. J. Gertler, "Survey of Model-Based Failure Detection and Isolation in Complex Plants", *IEEE Control Systems Magazine*, vol. 8, no. 6, pp. 3–11, 1988.

[18] A. S. Willsky, "A Survey of Design Methods for Failure Detection in Dynamic Systems", *Automatica*, vol. 12, pp. 601–611, 1976.

[19] D. M. Himmelblau, "Fault Detection and Diagnosis in Chemical and Petrochemical Processes", *Chemical Engineering Monograph 8*, 1978.

[20] O. Moseler and R. Isermann, "Applications of Model-Based Fault Detection to a Brushless DC Motor", *Industrial Electronics, IEEE*, vol. 47, no. 5, pp. 1015–1020, 2000.

[21] R. Isermann, "Model-Based Fault-Detection and Diagnosis – Status and Applications", *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, Jan. 2005.

[22] P. M. Frankt and X. Ding, "Survey of Robust Residual Generation and Evaluation Methods in Observer-Based Fault Detection Systems", *Journal of Process Control*, vol. 7, no. 6, pp. 403–424, 1996.

[23] P. M. Frankt, “Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy A Survey and Some New Results”, Survey Paper, *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[24] H. OCAK and K. A. Loparo, “A New Bearing Fault Detection and Diagnosis Scheme based on Hidden Markov Modeling of Vibration Signals”, Proceeding of the IEEE International Conference on Acoustics, Speech, and Signal Processing, May 2001.

[25] R. Isermann and P. Balle, “Trends in Applications of Model Based Fault Detection and Diagnosis of Technical Processes”, *Control Engineering Practice*, vol. 5, no. 5, p. 709–719, 1997.

[26] D. Cho and P. Paolella, “Model-Based Failure Detection and Isolation of Automotive Powertrain Systems”, Proceeding of the American Control Conference, p. 2898–2907, San Diego, 1990.

[27] J. J. Gertler, M. Costin, X. Fang, and Z. Kowalczuk, “Model-Based Diagnosis for Automotive Engines - Algorithm Development and Testing on Production Vehicles”, *Control Systems Technology, IEEE Transaction*, vol. 3, no. 1, pp. 61–69, 1995.

[28] A. Dutka, H. Javaherian, and M. J. Grimble, “Model-Based Engine Fault Detection and Isolation”, Proceeding of the American Control Conference, p. 4593–4600, St. Louis, 2009.

[29] S. Simani “Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques”, Doctoral Dissertation, University of Modena and Reggio Emilia, 2003.

[30] R. P. Lippmann, “An Introduction to Computing with Neural Nets,” *IEEE ASSP Magasine*, vol. 3, no. 4, pp. 4–22, 1987.

[31] D. Boswell, “Introduction to Support Vector Machines”, http://dustwell.com/PastWork/IntroToSVM.pdf, 2002.

[32] R. Burbidge and B. Buxton, “An Introduction to Support Vector Machines for Data Mining”, Research Paper, University College, London, 2008.

[33] C. J. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” *Data Mining and Knowledge discovery*, vol. 2, pp. 121–167, 1998.

[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Leaning: Data Mining, Inference, and Prediction*, Second Edition, Springer, 2008.

[35] C. R. Rao, "The Utilization of Multiple Measurements in Problems of Biological Classification", *Journal of the Royal Statistical Society*, vol. 10, no. 2, p. 159–203, 1948.

[36] W. J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*, Revised, Oxford University Press, 2000.

[37] M. Welling, "Fisher Linear Discriminant Analysis," *Science*, vol. 1, no. 2, pp. 1–3, 2009.

[38] T. Hastie, A. Buja, and R. Tibshirani, "Penalized Disriminant Analysis", *The Annals of Statistics*, vol. 23, no. 1, pp. 73–102, 1995.

[39] J. Mohammadpour, K. Grigoriadis, M. Franchek, and B. J. Zwissler, "Real-Time Diagnosis of the Exhaust Recirculation in Diesel Engines Using Least-Squares Parameter Estimation", *Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 1, p. 011009-01–08, 2010.

[40] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, "Applied Linear Statistical Models", Fifth Edition, Tata McGraw Hill, 2005.

[41] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, Apr. 2005.

[42] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society (B)*, vol. 58, no. 1, pp. 267–288, 1996.

[43] M. Caliendo and S. Kopeinig, "Some practical guidance for the implementation of propensity score matching", Working Paper, IZA Discussion Papers, No. 1588, Leibniz Centre for Economics, 2005.

[44] L. Clemmensen, T. Hastie, and B. Ersboll, "SDA Matlab package"

[45] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in *International Joint Conference on Artificial Intelligence*, 1995.

[46] P. Shook, "Steady finder r1v2.", Heavy-Duty Engine Performance Group, 2010.

APPENDICES

A: How ‘Representative’ is the Training Data?

Literature on data classification emphasizes the use of diverse training data for making a more robust, pragmatic model. Diverse here implies a more representative subset of the overall data. In the case of a field-tested diesel engine, a good representation would be data that embodies a considerable range of operating states, duty cycles, performance parameters and ambient conditions. This easily extends from the fact that the training data is the base knowledge of the classifier. More focused data that is less representative is bound to suffer in decision making and negatively affect the overall performance of classification.

During the course of this work, many different techniques were devised to pick the most suitable data for classification. Besides the usual filters that select the highway operational data (Section 4.2), selection criteria to improve matching quality, steadiness etc. were devised as well. One way to gauge the goodness of these criteria was the overall accuracy of the classifier that they led to. Another way could be to check the quality of training data selected, in terms of the representativeness of the data, as described above, and this can be achieved by inspecting the speed-load maps. A speed-load map is generated for every set of overall data by plotting net engine torque vs. speed. This is typically done for a day’s worth of data with the truck having travelled for sufficient miles. A speed-load map (Figure A.1: Speed-load map for T 103. offers great insight into the daily truck operation by clearly manifesting the more frequent operating regimes. The technique is simple and effective to make generalized observations.

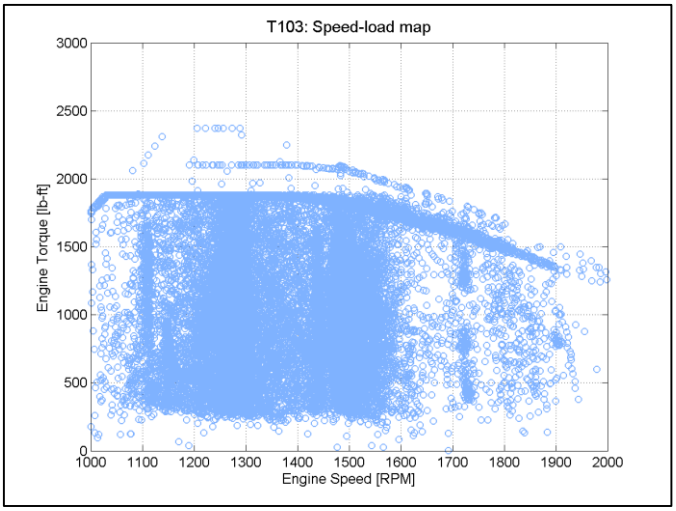

Figure A.1: Speed-load map for T 103.

Steady State Data Representation

Section 4.7 covers the performance of the steady state classification. It is of interest to see what sort of operating ranges the steady state data represents. T103 is taken as a single case for illustrative purposes. The data representation can obviously be studied for the other trucks in a similar fashion.

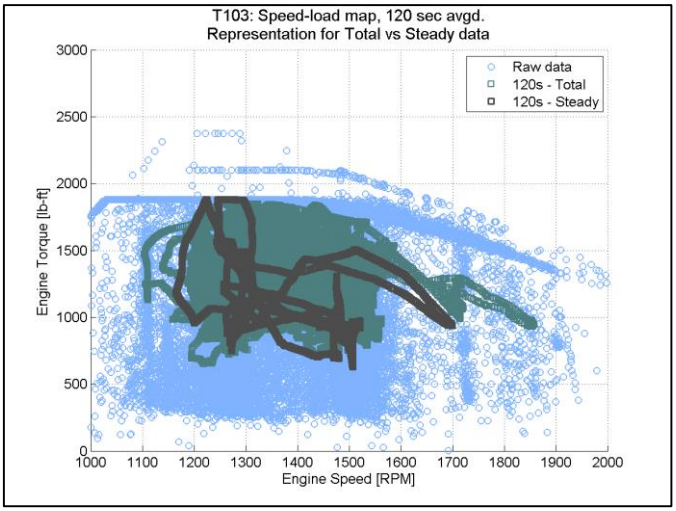

Figure A.2: Data representation for a moving averaged set of T 103 comparing steady vs. total data's operating regions.

It is seen from Figure A.2 that even though the steady data is much smaller in size, its spread, and so its representation is quite comparable to the total data. This may not be the case every time since a lot depends on the duty cycle and ambient conditions. There may not be a clear correlation between the data representation and performance of the classification since many other parameters are involved in finding the projections of data. However, this picture offers a good insight in the kind of data one can or may want to start with. A similarly interesting picture is the one that plots speed vs. torque for matching quality methods.

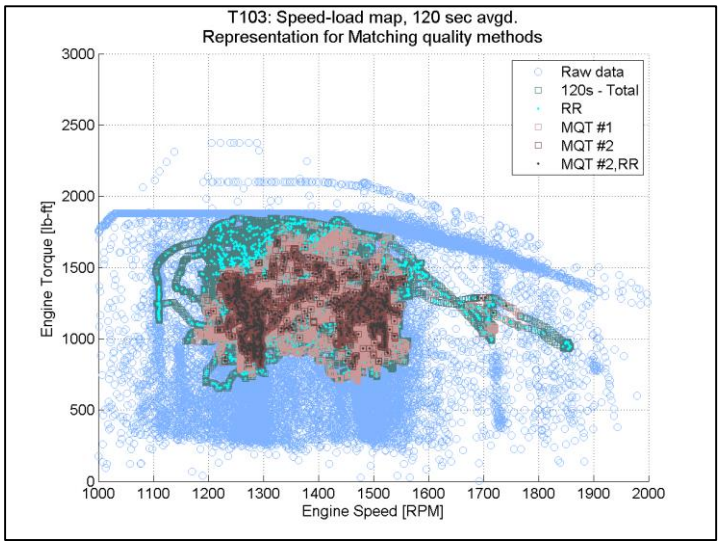

Figure A.3: Data representation of matching quality methods for T 103.

The blue backdrop is the speed-load map of the raw data. 120 s – Total is the 120 s moving averaged data with basic filtering. RR, MQT #1, MQT #2, and MQT #2 and RR denote the four matching quality methods discussed in section 4.5. Just as the steady data map, insights can be sought from this plot over the spread of data of each method.

Raw numbers are also useful to quantify the graphic. Table A.1 provides a summary of the number of points one starts with for every truck and the number that is left after applying different of data selection methods. The attrition of the number gives a great idea of the stringency of selection criteria and a need to change them, if any.

Table A.1: Numerical summary of all the data selection methods.

	Data Selection Methods					
Trucks	**120 s, moving avgd.**	**Steady data**	**RR**	**MQT #1**	**MQT #2**	**MQT #2, RR**
T103	34922	4415	7943	12075	3672	1380
T231	20686	5291	6960	852	389	243
T8572	32260	3788	9603	3677	3783	1702

B: Matlab Programs

```
%matchquality.m
%By Aniket Vagha

%This program uses a switch to present two cases of
achieving improvements in matching quality. Case I uses
absolute differences between healthy and faulty within a
pre-defined threshold. Case II uses modified standard
scores to chose appropriate indices of data.
switch mqsel %Switch caller
    case 1

        % Find common indices that satisfy absolute
        differences.
        idx = find(abs(DIFF(1,:))<50 & abs(DIFF(2,:))<10 &
abs(DIFF(3,:))<1
       & abs(DIFF(4,:))<0.5 & abs(DIFF(1,:))<50);

        % Data selection based on the indices obtained
        XF = XF(:,idx); healthymatch = healthymatch(:,idx);
        closestloc = closestloc(idx,:);
        DIFF = healthymatch - XF;

    case 2

        %Defining (x - mu) and standard deviations to
         calculate standard scores
        del = DIFF((1:5),:);
        sig = std(del,0,2);
```

```
        c1s = (del(1,:)./sig(1)).^2;
        c2s = (del(2,:)./sig(2)).^2;
        c3s = (del(3,:)./sig(3)).^2;
        c4s = (del(4,:)./sig(4)).^2;
        c5s = (del(5,:)./sig(5)).^2;

        % Generating a common metric: root-square
        metric = sqrt(c1s + c2s + c3s + c4s + c5s);
        idx = find(metric < sqrt(1.0));

        % Data selection
        XF = XF(:,idx); healthymatch = healthymatch(:,idx);
        closestloc = closestloc(idx,:);
        DIFF = healthymatch - XF;

        clear c1s c2s c3s c4s c5s del idx


end

%ThreeFault_Threeclass test set generator
%By Aniket Vagha

%Generation of an unbiased test set from the raw data. To
be used as a script in main program that extracts the
results. Even before defining training, we keep aside a
portion as Test Data. Since 4 different ways of matching
are to be tested, we pick 200 points each from healthy and
faulty raw data to make up the test data.
```

```
clear
clc
close all

% Defining 'datapm' as containing the names of the
datasets. Useful to directly address the cell array.

datapm =
{'T103_Rank_Analysis_total_fueling','T231_Rank_Analysis_tot
al_fueling','T8572_Rank_Analysis_total_fueling'}; % data
post matching - but raw.

nooffiles = length(datapm);

var2 = 6:26;

load('randidx.mat')
%randidx.mat is a .mat file that stores randomly generated
indices and is called here to use the indices for
extracting test sets.

trainidx = 1:19650; %Picking only 19650 points from every
truck

For j = 1:nooffiles

      load(datapm{j},'XH','XF')
      if strcmp(datapm{j}(1:5),'T8572')
          eval(['h'  datapm{j}(1:5) ' = XH' ';']);
          eval(['f'  datapm{j}(1:5) ' = XF' ';']);
      else
```

```
            eval(['h'  datapm{j}(1:4) ' = XH' ';']);
            eval(['f'  datapm{j}(1:4) ' = XF' ';']);
        end

end

%training set generation
train_healthy =[hT103(var2,trainidx) hT231(var2,trainidx)
hT8572(var2,trainidx)]';
faulty103train = fT103(var2,trainidx)';
faulty231train = fT231(var2,trainidx)';
faulty8572train = fT8572(var2,trainidx)';
train_size = size(train_healthy,1)/3;

% target train/group matrix generation
target_train_healthy= zeros(3*train_size,1);
zerotrain = zeros(train_size,1);
onestrain = ones(train_size,1); %Arrange these two to get
the coding
twostrain = 2*ones(train_size,1);
threetrain = 3*ones(train_size,1);

%concat sets generation
concat_train= [train_healthy; faulty103train;
faulty231train; faulty8572train];
ntruck = 3;

concat_target_train= [target_train_healthy; onestrain;
twostrain; threetrain];

%Creating Dummy Matrices
```

```
dummy(:,1)= [onestrain;onestrain;onestrain;
zerotrain;zerotrain;zerotrain];
dummy(:,2)= [zerotrain;zerotrain;zerotrain;
onestrain;zerotrain;zerotrain];
dummy(:,3)= [zerotrain;zerotrain;zerotrain;
zerotrain;onestrain;zerotrain];
dummy(:,4)= [zerotrain;zerotrain;zerotrain;
zerotrain;zerotrain;onestrain];

%Creating concatenated healthy test data
test_healthy =[hT103(var2,rid103) hT231(var2,rid231)
hT8572(var2,rid8572) ]';
test_size = length(rid103);

faulty103test = fT103(var2,rid103)';
faulty231test = fT231(var2,rid231)';
faulty8572test = fT8572(var2,rid8572)';

target_test_healthy = zeros(3*test_size,1);


zerotest = zeros(test_size,1);
onestest = ones(test_size,1); %For encoding
twostest = 2*ones(test_size,1);
threetest = 3*ones(test_size,1);

%Concatenated test and its encoding
concat_test = [test_healthy; faulty103test; faulty231test;
faulty8572test];
concat_target_test = [target_test_healthy; onestest;
twostest; threetest];
```

```
%clear redundant variables
clearvars -except test_healthy test_size
target_test_healthy train_healthy train_size
target_train_healthy concat_test Concat_target_test
concat_train concat_target_train dummy

% sldafcn.m
%By Aniket Vagha
%This program uses the training and test selection from the
previous routines and carries out the Discriminant Analysis
and classification. It passes on the error rates to its
parent for plotting and data representation.

        %Load data and other misc steps, concatenate it.
        concat_steady= [concat_train; concat_test];
        [n,~]= size(concat_steady);

        train_size = 6*train_size;%Multiply by 4 for two
        fault, 6 for three fault.
        Itr= 1:1:train_size;
        Iout=train_size+1:n;

        %Concatenated group matrix
        Yclass= [concat_target_train; concat_target_test];

        Xtr= concat_train;
        Ytr = concat_target_train;


        [Xtr, mx, vx]= normalize(Xtr);%Normalize Data
```

```
In= find(vx>sqrt(eps)); %Find variables that truly
change
Xtr= Xtr(:,In); %Extract variables that are non-
constant

%Center data around the training data set mean
concat_steady= (concat_steady(:,In)-
      ones(n,1)*mx(In))./sqrt(ones(n,1)*vx(In));


%Set Parameters
lambda = 0;% Ridge reg coefficient
prior= [3 1 1 1]; % Refer 'priors' in Chapter 6.
maxiter = 50; % max iteration in SDCA algorithm.


tic %Initiate time and other parameters.
len = length(Xtr(1,:));
e_tr = zeros(1,len);
e_tst = zeros(1,len);

 for i= 1:len;
       stop = -i; % l1-norm. negative: number of
   vars in LARS-EN


    %Perform SLDA

    [sl,~,~]=
slda(Xtr,dummy,lambda,stop,maxiter,0);
```

```
            %Project data along Sparse directions
            DC= concat_steady*sl;

            %Classification of training set
            [class_tr,~,~]= classify(DC(Itr,:), DC(Itr,:),
                          Yclass(Itr),'linear', prior);
            trerr =
            length(find(class_tr~=Yclass(Itr))) /
                     length(Yclass(Itr))*100;
            e_tr(i)= trerr;

            %Classification of test set
            [class_tst,~,~]= classify(DC(Iout,:),DC(Itr,:),
                          Yclass(Itr),'linear', prior);
            tsterr = ( length(find(class_tst~=Yclass(Iout))
                      /length(Yclass(Iout)))*100;
            e_tst(i)= tsterr;

        end
```

```
ThreeFault_ThreeClass Results generator
% By Aniket Vagha

%This scripts calls in TestGen_ThreeFault_ThreeClass and
sldafcn. Basically using these two sub-routines, this
program performs Discriminant Analysis and classification
using classify.m. The main motive is result extraction
%in the form of plots and data tables.
clear
close all
clc
```

```
%Do not use the same code for steady, rem repeats etc. Mak
copies and
%rename acc.


        Chap6_C1_TestGen_ThreeFault_ThreeClass %Test data
generator
        sldafcn % SLDA - classification
        etr1 = e_tr; etst1 = e_tst; %Storing the training
and test errors resp.

       figure(1);
            %Figure 1 makes two subplots, one for the
training error rate and the other
            %for the test error rate. Axes and Figure
properties are used
            %for aesthetics and clarity.
            subplot(10,2,(1:2))
            set(gca,'Visible','off','Units','normalized');
            titline = text(0.45,0.95,{'T103, T231,and
          T8572-';'in three-fault binary classification'});

set(titline,'FontName','Arial','FontSize',14,'HorizontalAli
gnment','Center');

            subplot(10,2,(5:2:19));
            plot(etr1,'-c','Marker','o','MarkerSize',7,
'MarkerFaceColor','c','MarkerEdgeColor','k','linewidth',4);
```

```
            YLabel = ylabel('Percentage Error [%]');
            catitle1 = title({'Training error';'19650 pts
            per truck per class'});
            ylim([0 50]);
            xlim([1 27]);
            grid on
            set(YLabel,'FontSize',13)
            set(catitle1,'FontName','Arial','FontSize',13)
            set(gca,'FontSize',13)
            xlabel1 = text(21,-5,'Number of variables /
            stops used [no.]');
            set(xlabel1,'FontName','Arial','FontSize',13)

            subplot(10,2,(6:2:20))

            plot(etst1,'-c','Marker','o','MarkerSize', 7,
'MarkerFaceColor','c','MarkerEdgeColor','k','linewidth',4);

            catitle2 = title({'Test error';'200 pts per
            truck per class'});
            ylim([0 50]);
            xlim([1 27]);
            grid on
            set(catitle2,'FontSize',13)
            set(gca,'YTickLabel',[],'FontSize',13)


        figh = figure(2);
        %This is subplot for instances of misclassification
in these three trucks.
```

```
            subplot(10,2,(1:2))
            set(gca,'Visible','off','Units','normalized');
            titline = text(0.45,0.95,{'T103, T231, & T8572
            - ';'Instances of misclassification'});

set(titline,'FontName','Arial','FontSize',14,'HorizontalAli
gnment','Center');

            subplot(10,2,(5:2:19));
            plot(Ytr,'-mo','linewidth',3);
            hold on;
            plot(class_tr,'c');
            catitle1 = title('Training Data');
            ylim([-0.05 3.05]);

            grid on
            set(catitle1,'FontName','Arial','FontSize',13)

set(gca,'FontSize',11,'YTick',(0:3),'YTickLabel',{'0:Health
y'; '1: T103 Faulty';'2:T231 Faulty';'3:T8572 Faulty'})
            xlabel1 = xlabel('time [sec]');
            set(xlabel1,'FontName','Arial','FontSize',13)

            subplot(10,2,(6:2:20))
            plot(concat_target_test,'-mo','linewidth',3);
            hold on;
            plot(class_tst,'c');
            catitle2 = title('Test data');
            ylim([-0.05 3.05]);

            grid on;
```

```
            set(catitle2,'FontName','Arial','FontSize',13)

set(gca,'FontSize',13,'YTick',[0,3],'YTickLabel',[])
            xlabel1 = xlabel('time [sec]');
            set(xlabel1,'FontName','Arial','FontSize',13)

            %SAVE THE PLOTS!!
```